University of Alberta

ACTIVELY SEARCHING CONTOUR MODELS FOR CONTOUR EXTRACTION

by

Jiankang Wang ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2000

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59691-5

Canada

University of Alberta

Library Release Form

Name of Author: Jiankang Wang

Title of Thesis: Actively Searching Contour Models for Contour Extraction

Degree: Doctor of Philosophy

Year this Degree Granted: 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Jiankang Wang

Jiankang Wang
8-1008, 41 Wai-Guan Street
An-Wai, Beijing
P. R. China, 100011

Date: Sept. 29, 2000

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Actively Searching Contour Models for Contour Extraction** submitted by Jiankang Wang in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Xiaob Li
Dr. Xiaobo Li

Dr. Herbert Yang

DG Elliott
Dr. Duncan Elliott

Dr. Hong Zhang

Dr. John Buchanan

Date: Sep. 29, 2000

Abstract

We present a novel system, called Actively Searching Contour Models (ASCMs), for contour extraction. ASCMs are based on two variants of the Active Contour Models (ACMs): balloons and ziplock snakes. Unlike the traditional ACMs, ASCMs are designed to *actively* search for desired contours, as opposed to being passively attracted to image features. ASCMs start from trivial initializations and expand or shrink to find the right contour, and, thus, they can be used in automatic applications.

Our ASCMs are able to integrate other useful information sources, in addition to raw image features such as intensity and edges. First, mid-level image features produced by a bottom-up edge linking-like method are effectively used. The bottom-up method uses local information to reduce noise and re-enforce detected structures and, thus, produces reliable mid-level features. Second, information collected during the searching process is used to predict and/or confirm searching directions. Finally, shape-models, which possess high-level information, are adopted to direct the searching process and to provide an informed guess.

To make the searching process more controllable, we begin to improve the balloons by re-modeling them as a series of spring-operated rods. The resulting model is less affected by the peeling problem which we observed in applying balloons to noisy images. Subsequently, we propose a multi-scale, non-shrinking internal energy model, which is also unbiased towards expanding or contracting. The multi-scale effect is achieved by combining multiple-snakes, and this model provides a balloon with a controllable smoothness constraint, which is essential for contour completion.

To accurately search for contours, we then propose using the Saliency Map method, and subsequent steps, to build a reliable potential field from an edge image. We also present an adaptive inflating force which can drive a balloon to converge to edges with a pre-specified strength.

The above methods are then integrated using the backtracking strategy, which performs a multi-level feature search. Guidelines for parameter choosing are also discussed. Numerous experiments are conducted, and the results show that the ASCMs are controllable, accurate, and easy to adjust.

To my loving parents
and my dear wife

Acknowledgements

I would like to thank my supervisor, Dr. Xiaobo Li, for guiding me through the research with insight and patience. I am also grateful for his confidence in my capability and for his support in various ways.

I would also like to thank my thesis examining committee members, Dr. Hong Zhang, Dr. John Buchanan, Dr. Herbert Yang, and Dr. Duncan Elliott for carefully reading the thesis and providing numerous suggestions that helped in improving it.

This thesis is dedicated to my family: my parents for loving me and supporting me in various ways; my wife, Xiaojuan, for her love and unlimited patience; and my daughter, Catherine, for the joy she has brought to my life.

Finally, I would like to thank all my friends, especially Xinguang Chen and Guohu Huang, for their friendship. Many thanks to Edith Drummond, Anne Nield, Carol Smith, Sunrose Ko, and other Computing Science staff for their help. I also want to thank all Edmontonians for their hospitality.

Contents

1 Introduction **1**
- 1.1 Contour Extraction: A Review 1
  - 1.1.1 Bottom-up Methods 2
  - 1.1.2 Top-down Methods 3
- 1.2 Why Active Contour Models? 7
- 1.3 Active Search and ACMs 8
- 1.4 Overview of the Thesis 9
  - 1.4.1 Objectives of the Thesis 9
  - 1.4.2 Contributions of the Thesis 9

2 Semi-Rigid Snake Models **13**
- 2.1 Introduction: The Peeling Problem 13
- 2.2 Dynamic Strong Snake Parts Identification 17
- 2.3 Computing Influencing Forces 19
- 2.4 Implementation and Experiments 21
- 2.5 Conclusions 26

3 New Internal Energy Models **34**
- 3.1 Problems with the Balloon Models 34
- 3.2 An Unbiased Non-Shrinking Model 35
- 3.3 Maintaining Tension 38
- 3.4 A Multi-Scale Internal Energy Model 40
  - 3.4.1 Smoothness Constraint and Contour Completion 40
  - 3.4.2 Achieving Multi-Scale via Multiple Snakes 41
- 3.5 Experiments 43
- 3.6 Conclusions 45

4 Content Guided Search **53**
- 4.1 Introduction 53
  - 4.1.1 Method Overview 54
- 4.2 Tracing Curves 55
- 4.3 Content-Guided Search 57
- 4.4 Experiments 58
- 4.5 Conclusions 63

5 Controlled Accurate Search: A Combined Model **67**
- 5.1 Introduction 67
- 5.2 Building up External Force Fields 68
  - 5.2.1 Rectifying Edge Magnitudes 68
  - 5.2.2 Non-Maximum Suppression 70
- 5.3 Adaptive Inflating Forces 72
  - 5.3.1 Estimating Image Forces 76
  - 5.3.2 Backtracking 77

5.4 A Combined Model 78
5.5 Experiments 80
5.6 Conclusions 82

6 Shape-Model Guided Search **96**
6.1 Introduction 96
6.2 Encoding *a priori* Information 97
6.3 How Ziplock Snakes Work 98
6.4 Combining Ziplock Snakes with the Grammatical Model 99
  6.4.1 Competing Interpretations 100
  6.4.2 Predicting the Next Step 102
  6.4.3 Further Reducing Computation 104
6.5 Implementation and Experiments 105
6.6 Conclusions 107

7 Conclusions and Future Work **111**
7.1 Conclusions 111
7.2 Future Work 112

A Dynamic Programming for Energy Minimization of Balloons **113**

B Proof of Snake Properties **116**

Bibliography **118**

List of Figures

1.1 The relationship between ASCMs and the segmentation methods in the literature. 10
1.2 An overview of ASCMs: components that comprise the whole system. 12

2.1 An illustration of the "peeling" problem: (a) a synthetic boundary; (b) a probable intermediate state of the balloon; (c) snaxel x keeps moving outward; d) snaxel y is pulled away from its equilibrium location. 14
2.2 How the peeling problem is related to the internal energy computation method: (a-c) three consecutive states of a snake; (d) for snaxel x, a small neighborhood (a 3x3 grid) is considered when its energy is minimized. 15
2.3 A spring-operated hinge is a more accurate model of a spline. 15
2.4 A simulation configuration: snaxels y_{1-4} are not movable while x_{1-3}'s positions are determined by the internal energy and the inflating force. 16
2.5 The internal energy of y_2 during iteration modeled as a string or a rod with spring-operated hinges. 16
2.6 An illustration for influence-measure computation: (a) influence measures for snaxel x_i are based on x_i's neighboring parts on a snake, namely a and b. (b) angles used in Eqns 2.1 and 2.2 18
2.7 Pseudo codes for computing influence measures. 19
2.8 An example of influence measures: (a) a synthetic boundary with noise; (b) an example of I_p in which the size of the snaxels indicates the magnitude; (c) the corresponding I_s. 20
2.9 An example of generating influencing forces: (a) part a is extended in the tangent direction; (b) extended by maintaining the internal angle; (c) curve-extending is done from both directions. 21
2.10 Converting forces into potential energies (Eqn. 2.4). 22
2.11 An example using a synthetic line-drawing image: (a) the test image with an initialization; (b) the result produced by the semi-rigid balloon models with a normal force of 0.35; (c) and (d) are the results produced by the original balloons with normal forces set to 0.35 and 0.32 respectively. 23
2.12 The semi-rigid balloons can cope with gaps on boundaries: (a) the test image; (b) the final result. 24
2.13 For some bad initializations (e.g. (a)), the semi-rigid balloons can recover from the error ((b), the snake after a few iterations). 24
2.14 A cup image example: (a) an initialization; (b) a result using the semi-rigid models; (c) & (d) results using the original balloons. 25
2.15 Two test-image examples with a noise level of 0.40 and 0.80, respectively. The circle has a radius of 50 and four gaps 30 pixels in length. 25

2.16 A test on when the peeling effect occurs. The test image has a gap size of 30 and a noise level of 0.0. The semi-rigid models perform consistently better than the original balloons and can reach a stable state even with a large normal force. 28
2.17 Another test on when the peeling effect occurs. Fig. 2.17 shows the same test image except that the noise level is now increased to 0.40. With the noise level increased, the normal force has to be larger than a certain threshold (0.3) for the balloons to perform well. 29
2.18 Effects of the force scaling factor κ. The test image has a gap size of 30 and a noise level of 0.4. Once the normal force is larger than a certain threshold (0.30 in this case, related to the noise level), the semi-rigid models perform best with a medium κ (2.0). 30
2.19 Effects of the parameter ρ. Gap size = 30, and noise level = 0.4. The models perform best with a medium value (1.5 – 2.0). 31
2.20 The semi-rigid models perform consistently better than the original balloons. Even when both of them cannot reach a stable state (noise level ≥ 0.8), the semi-rigid models produce results with a much smaller MSE. The gap size = 30. 32
2.21 Effects of parameter δ. The gap size is 30, and the normal force = 0.35. When $\delta > 0.4$, the algorithm performs well consistently. 33

3.1 A new smoothness constraint: (a) the internal angle ϕ; (b) the smooth area (enclosed by the two arcs) and the two force vectors computed to pull snaxel z or z' to the smooth area; (c) another force vector $\mathbf{w}$ generated to maintain the equal distance property. 36
3.2 An example using the new smoothness constraint: (a) an initialization; (b) the final shape after energy minimization with only the internal energies. 37
3.3 Two vectors $\mathbf{v}$ and $\mathbf{w}$ for tension computation. They are computed only when $|y-z|$ and $|x-z|$ are larger than the average distance h. . 38
3.4 (a) under the influence of internal energies, the next state of a balloon initialized as a circle is a larger concentric circle; (b) a typical tension change pattern. 39
3.5 A snake discretized with different scales: it is considered (a) smooth using a small scale; (b) not smooth using a larger one. 40
3.6 A simulation configuration: (a), (c), and (e) show three partial balloons with snaxels $a-d$ fixed; (b), (d), and (f) show their final shapes under the same inflating force. Arrows indicate the inflating forces applied to these partial balloons. 41
3.7 The average tension and smoothness energies for the three partial balloons shown in Fig. 3.6. 42
3.8 An example of a multi-scale snake. Three component snakes are shown in this configuration. The second and third snakes are attached to the first one with virtual strings. 43
3.9 e_{FD} plotted against the gap size. The noise level is 0.0. Squares indicate non-stable solutions. 44
3.10 The test images are the same as those used in the first experiment, except they are now added with zero-mean Gaussian noise with a standard deviation 0.40. All four performance measures are plotted against the gap size. The multi-scale models can produce stable solutions all the time. Continued in Fig. 3.11. 47
3.11 Continued from Fig. 3.10 48

3.12 The gap size is fixed at 40, but the noise level changes from 0.0 to 0.8. All four performance measures are plotted against the noise level. Continued in Fig. 3.13. 49
3.13 Continued from Fig. 3.12. Figure (a) shows that the rigidization process performs equally well for both methods. The differences in other performance measures account for the performance difference of the two methods. 50
3.14 (a) a part of a balloon trapped by a local minimum; (b) forces shown as arrows are generated from the two component snakes. 51
3.15 e_{FD} computed during iteration of a multi-scale snake with two component snakes, and a single scale snake. A smaller e_{FD} indicates that the balloon is smoother during the iteration. 51
3.16 (a) a circle with numerous small gaps and with noise level 0.4. A typical result produced (b) by a single scale snake; (c) by a multi-scale snake with two component snakes. Note that the single scale snake has the same average distance as the second snake of the multi-scale snake. 52
3.17 (a) a contour with a gap and a convex part. (b) the result produced by a multi-scale snake. Both the gap and the opening to the convex part are treated as gaps, which is not true for the convex part. 52

4.1 An illustration of content guided contour search: Arrows indicate forces generated from local image content. 55
4.2 A saliency map example: (a) a contour with a gap; (b) its computed saliency map. 56
4.3 Tracing curves: (a) c-a is the most salient curve that originates from c; (b) the starting point of the second most salient curve is picked from a set of 7 points at the opposite side of the first starting point. 57
4.4 Pseudo codes for computing angles between a snake and a traced curve. The angles are then used to verify the curve's suitability for content guiding. 59
4.5 (a) a series of internal angles are computed as part of curve goodness measures; (b) several force vectors are generated to perform content guiding. 60
4.6 Some typical shapes with different initializations. 61
4.7 A bad initialization example: (a) an initialization; (b) the result produced by the content-guided snake model. 62
4.8 (a) a contour with various small gaps; (b) the contour is successfully extracted by a content guided balloon. 62
4.9 Dealing with irregular shapes: (a) an irregular shape with an initialization; (b-c) two typical results produced by the original balloon models with different normal forces; (d) result produced by the content guided balloons with either normal force. 63
4.10 (a) an example test image with a noise level of 1.2; (b) the input edge image to the SM method; (c) the produced saliency map. 64
4.11 Testing the noise-resisting property of the content-guided model. The test image has a unit contrast circle with zero mean Gaussian noise. Noise level in this figure stands for the standard deviation of the Gaussian noise. 65
4.12 (a) a contour with both a bump and a gap; (b) the input image to the SM method; (c) the saliency map produced by the SM method; (d) the contour successfully extracted by the content-guided model. . . . 66

5.1 (a) an ultrasound image; (b) an edge image produced by a LoG operator with $\sigma = 1.0$. 69

5.2 (a) the edge image produced by a Sobel operator; (b) the saliency map with $\rho = 0.5$ and $N = 20$; (c) the improved saliency map. 69
5.3 The potential energy related to the step edge in (a) is h, where that of the smoothed edge in (b) is h'. 70
5.4 (a) a contour with unit intensity and a noise level of 0.4; (b) the saliency map; (c) after the AND operation; (d) after non-maximum suppression (tracing). 72
5.5 (a) another example with noise level 0.8; (b) the saliency map; (c) after the AND operation; (d) after tracing. 73
5.6 Results for the ultrasound image shown in Fig. 5.2: (a) after the AND operation; (b) after tracing. 74
5.7 Using a constant inflating force: (a) the image force, f_{image}, is the only external force; (b) a constant inflating force as large as the noise threshold, t_{edge}; (c) the composite external force. Shaded areas indicate that the composite force is too large or too small. 75
5.8 An adaptive inflating force: (a) the expected final external force after compensating for the insufficient part and removing the excessive part; (b) the adaptive inflating force plotted against the underlying image force. 76
5.9 Estimating image forces for adaptive inflating force computation from a 3x3 neighborhood. 77
5.10 An illustration of the backtracking mechanism. 78
5.11 A diagram illustrating contour extraction using the Actively Searching Contour Models. 79
5.12 Testing the effect of t_{edge} and $t_{tension}$. The test image is a circle (radius 50) with four gaps of size 30. 84
5.13 Comparing an adaptive balloon with the traditional balloon. The parameter, $t_{tension}$, for the adaptive balloon is set to 0.15. 85
5.14 An experiment on parameter sensitivity: (a) a contour with a gap; (b) the result using the adaptive balloon; (c) and (d) results of the traditional balloon with different inflating forces. It should be noted that the balloon shown in (d) is not able to reach a stable state. . . . 86
5.15 (a) a test image for accuracy testing. The third circle from inside has a radius of 50; (b) a plot of the extracted circles' intensity levels against the parameter t_{edge}. 87
5.16 (a) multiple concentric circles with gaps; (b) a result using the traditional balloon with a constant force of 0.35; (c) and (d) results of the adaptive balloon with the inflating force set to 0.35 and 0.41, respectively. 88
5.17 (a) a contour with some weak parts; (b) the first equilibrium state of an adaptive balloon; (c) the final result after several cycles of backtracking. 89
5.18 (a) a square with sharp corners; (b) a balloon with multi-scale and semi-rigid modules is not able to reach corners; (c) the balloon converges to the corners after disabling the two modules in the second sub-step. 89
5.19 (a) a contour with a large opening; (b) and (c) show the first two equilibrium states of the backtracking process; (d) the final state, which is a better regularized solution. 90
5.20 (a) a loin image; (b) its saliency map; (c) and (d) the first and final states. 91
5.21 Another loin image example: (a)-(d) the loin image, its saliency map, and the first and last equilibrium states. 92

5.22 The third loin image example: (a)-(d) the loin image, its saliency map, and the first and last states of a balloon. 93
5.23 The fourth loin image example: (a) the loin image; (b) the contour extracted. 93
5.24 A cup example: (a) the cup image; (b) its saliency map; (c) the adaptive result; (d) the result produced by the traditional balloon. 94
5.25 A CT medical image example: (a) the CT image; (b) its saliency map; (c) and (d) an initialization for a shrinking snake, and the corresponding result; (e) and (f) an initialization for an expanding balloon, and the result produced. 95

6.1 (a) a square; (b) rays and candidate points for the Olstad method; (c) a difficult case for choosing candidate points; (d) two user-selected starting points for a ziplock snake. 97
6.2 A nondeterministic finite-state machine for the regular expression that describes a rectangle. 98
6.3 An illustration of a ziplock snake. The head and tail are the active parts that are under the influence of the image potential. The force boundaries are moved gradually to convert more of the passive part into active part. 100
6.4 Difficult cases for ziplock snakes: (a) multiple contours crossing one another; (b) a single contour with gaps. 100
6.5 A binary tree listing all possible interpretations shorter than 6 for the regular expression $L^*RL^*RL^*RL^*R$. Solid lines indicate the paths to the five most possible interpretations. 101
6.6 A plot of a Gaussian distribution with $\mu = 100$ and $\sigma = 20$. The corresponding $P(L)$ and $1.0 - P(L)$ are also shown. 103
6.7 (a) an intermediate state of a ziplock snake; (b) all possible next steps for the right-side active segment; (c) all traced curves from the corresponding force boundary; (d) compatibility is mapped through an exponential function of the shaded area. 104
6.8 (a) a contour; (b) an intermediate state; (c) after some more iterations. 105
6.9 (a) multiple contours; (b) the square is extracted when given a description for a square; (c) the arch extracted when the shape model is for an arch. 106
6.10 (a) a square with some gaps; (b) the square extracted successfully. . . 106
6.11 (a) a square with unit contrast, and corrupted by zero-mean Gaussian noise with a standard deviation of 1.0; (b) the corresponding saliency map generated for curve tracing. (c) the ziplock snake deviates from the straight side line when it is attracted by noise; (d) the deviation is corrected once the snake locates strong curves; (e) the final results; (f) the darker area indicates the part of the image that is searched by the curve-tracing method. 108
6.12 (a) the same image as in Fig. 6.11 (a) except the noise standard deviation is increased to 1.2; (b) the corresponding saliency map; (c) the final result; (d) the area that is searched for possible next steps. . . . 109
6.13 (a) an image with several rectangles; (b) the corresponding saliency map; (c) the largest rectangle is extracted when the expected length and standard deviation of the side lines are set to 100 and 20 respectively; (d) the lower half of the largest rectangle is extracted when the expected length is lowered to 70. The initializations and other parameters are exactly the same. 110

A.1 The 3x3 grid (m=9) indicates the searching area for each snaxel. The number of combinations of possible moves involving 5 snaxels is m^5. Curved arrows show one possible move. 115
A.2 Converting the normal inflating force into potential energies. 115

List of Tables

2.1 Parameters used in the following examples unless otherwise specified. 22

4.1 Iteration numbers for examples shown in Fig. 4.6. 60

5.1 Average noise intensity values before and after pre-processing. 71
5.2 Average contour intensity values before and after pre-processing. . . . 74

6.1 The corresponding state table for the automaton shown in Fig. 6.2. μ and σ are the expected length and deviation of a certain graphical element such as an arc or a straight line denoted by the corresponding symbol. 99

Chapter 1

Introduction

Image segmentation is the first essential step in a computer vision system. It has been extensively investigated in the computer vision community. The following is a commonly accepted informal definition of image segmentation [25]:

> *Image segmentation is to divide an image into non-overlapped regions. Each region should be homogeneous and uniform with respect to some characteristic such as pixel intensity or texture.*

Hundreds of segmentation techniques are discussed in the literature. They are usually divided into three categories [20]: (1) characteristic feature thresholding or clustering, (2) edge-based methods, and (3) region-based methods. Different categorization frameworks [20, 27, 40, 43] exist because researchers look into these techniques from different perspectives. Usually one method works best for one specific imaging modality or application because general images have widely varying features in versatile applications.

The image segmentation problem is a so-called ill-posed problem, *i.e.* there is no unique and stable solutions to it. One of the reasons for this is that the image acquisition process, which projects 3D scenes into 2D images, induces information loss [6]. Computer vision researchers have been working to provide constraints to such kinds of ill-posed problems.

In this thesis, we focus on extracting contours from noisy images. Our method is based on the famous Active Contour Models (ACMs), or snakes [30, 31], but requires only a trivial initialization and can search for contours in a far larger area. Our snake model, termed Actively Searching Contour Models (ASCMs), can effectively utilize and combine various information sources such as raw image and shape models. Moreover, the behavior of our snake model is more predictable and controllable thanks to separated parameters for various desired functions.

1.1 Contour Extraction: A Review

Development of image segmentation techniques has been largely influenced by Marr's view of computer vision [35]. Under Marr's paradigm, recognition is done step by step, with each step relying only on the outputs of previous steps. This framework greatly simplifies recognition system design and has been widely accepted. Methods based on this view are usually called *"bottom-up"* methods in that information flows from pixels to features. A problem with this approach is that errors made in early steps propagate to later steps without opportunity for correction. An example in image segmentation is the two-step boundary extraction framework: edge detection followed by edge linking. Marr's framework has achieved considerable success but has by no means solved the problem.

The human visual system (HVS) can effectively recognize objects in 2D images. Some researchers point out that the HVS has incomparable features that computer vision systems do not have, such as large-scale parallel processing, multi-scale processing, and, more importantly, effective use of knowledge. Inspired by this insight, researchers proposed another category of method featuring exploitation of knowledge. Rule-based methods, scene labeling, deformable models, and active contour models [25, 31] are in this category and have achieved considerable success, too. These types of methods are usually called *"top-down"* methods since they start from a high-level scene model and find evidence at the pixel level to support the model. One point to note is although this categorization method is general purpose, we use it to categorize contour extraction methods only.

1.1.1 Bottom-up Methods

Bottom-up methods consist of two steps: edge detection and boundary formation. In the first step, discontinuities in the image are detected by a high pass filter or a gradient operator. The second step then deals with eliminating false edges and filling in gaps in boundaries. The second step is usually termed *edge-linking*.

Edge detection methods have been extensively investigated in the literature. Sobel, Roberts, Prewitt, and the Laplacian gradient operators [22] are all difference operators that respond to changes in gray levels or average gray levels. However, these gradient operators respond not only to edges but also to isolated points. Since they are either first or second difference operators, they are sensitive to noise and thus not directly suitable for processing noisy images. It is usual to have a smoothing step before applying difference operators to reduce the effects of noise.

By combining the smoothing step and difference operator, Marr and Hildreth proposed the Laplacian of Gaussian (LoG) operator [25]. It is normally denoted by $\nabla^2 G$; where the Laplacian is given by

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{1.1}$$

where f is the intensity image; and

$$G = e^{-(x^2+y^2)/(2\pi\sigma^2)} \tag{1.2}$$

is a two-dimensional Gaussian distribution with standard deviation σ. The Gaussian part of the LoG operator smoothes the image so that the Laplacian operator will not respond to structures smaller than σ. The Gaussian function is chosen because it has the desirable property of being smooth and localized in both the spatial and frequency domains. Also, since it is infinitely differentiable, the LoG operator is not an ill-posed operator.

Other similar methods include Torre and Poggio's regularization method [48], Haralick's facet model [26], and Canny's edge detector [9, 8]. Although researchers have used different kinds of frameworks, they have all derived filters similar to the Gaussian one. Use of the Gaussian filter was also confirmed by the results proposed by psychophysiologists and neurophysiologists indicating that there are physical analogies in the HVS [29].

Since the result of an edge detection operation is usually a set of spurious edges with many gaps, it is then necessary to have an edge aggregation process applied to the image after an edge detection operation. A graph-theory based pixel search method was used to extend discontinuous edge segments and to fill in gaps [36]. Nalwa and Pauchon [37] used a connectivity grid technique to map edge segments into connected graphs and then attempted to fill in the edge gaps. Zhu *et al.* [58] presented a method inspired by a technique called *potential function*, which originated in the field of physics. An edge image is modeled as a potential field with energy depositions at the detected edge positions. The energy of an edge pixel is influenced by its neighboring edge pixels. The algorithm connects discontinuous edge segments by evaluating energy at each edge pixel. Like other edge-linking methods, this method is also iterative and can fill only small gaps.

Sha'ashua and Ullman's [1, 44] saliency map was originally proposed to detect salient structures in line-drawing images. However, in [1], examples of detecting salient structures in gray-level images are shown. The method itself is similar to Zhu's potential function method [58] in concept; however, unlike others, the saliency map method provides a map of saliency values for all the image pixels instead of a map of edge points, *i.e.*, it indicates the strength of edges, not just their location.

1.1.2 Top-down Methods

The Hough transform [4] is the first method that tries to use a global model to detect object boundaries in noisy image data. It maps the edge elements from image space to a parametric space, where parameters of the mathematical equations that describe the object boundaries are extracted. The Hough transform is highly noise immune because parameters are decided by an accumulation process. However, the computational complexity can be prohibitive. Moreover, the requirement of a pre-specified algebraic form of the boundary equation is very strict. Ballard [4] extended the method to use models that cannot be described by an algebraic equation. However, this method still cannot deal with boundaries that are deformed, which is common in image segmentation.

Deformable models have been extensively used in boundary extraction. In [19], they are divided into two categories: (1) parametric models, and (2) pseudophysical models. A pseudophysical model can be envisioned as a non-rigid material influenced by image-derived forces. The problem is solved by using techniques developed in mechanics. Pseudophysical models are usually represented by splines or simply a set of discrete points. On the other hand, parametric models use parameterization of shape to describe a boundary. The parameters used are global in the sense that changing one parameter will influence the shape of the whole contour. Deformation and minimization are done in the parameter space.

Parametric Deformable Models

Staib *et al.* proposed a method [45] which uses the Fourier series to parameterize boundaries. *A priori* information is included as a spatial probability expressed through the likelihood of each model parameter. In [56], this model is used to locate boundaries in medical images. An initial guess is needed to put the boundary on a gradient image. The template model is then deformed to maximize an objective function in which gradient information is involved. In [10], information other than gradient, *e.g.* region information, is further incorporated and makes the boundary localization more accurate and reliable. In [46], the model is extended to deformable surfaces for segmenting 3D medical images. However the need for an initial guess makes it difficult for the method to be used in an automatic segmentation process.

Cootes *et al.* [15, 16] considered the Karhunen-Loeve (KL) decomposition of the global model observed on a training set of representative shapes. Through the use of a KL transformation, shapes are approximated by main variation modes extracted from the training data. As in Staib *et al.*'s method, the solution is sought by adjusting parameters to fit the model to edges extracted from an image. Gradient descent or relaxation methods are used to solve the problem, and again a close initialization is required.

In summary, parametric deformable models are powerful in their descriptive ability and make it easy to incorporate *a priori* information. Compared with the Active Contour Models (discussed in the next section), they do not have a smoothing term to regularize the solution, a property which is important when the image is noisy.

Pseudophysical Deformable Models

Active Contour Models were first proposed by Kass *et al.* [31] as a regularization approach to the ill-posed edge-detection problem. A snake is a smooth spline under the influence of image forces and other external constraint forces. The internal spline forces serve to impose a smoothness constraint on the detected edges. The image forces push the snake towards salient image features such as lines, edges, and subjective contours. A snake can be either closed or open, depending on whether the end points are connected.

The snake model is an analogy to mechanical systems, and influencing forces can be represented by equivalent potential and kinetic energies. Snake movement is governed by minimization of the total energy. Representing the position of a snake parametrically by $\mathbf{v}(s,t) = \mathbf{v}(x(s,t), y(s,t))$ where s and t are spatial index and time respectively, we can write its energy functional as

$$E_{snake} = \oint (E_{int}(\mathbf{v}) + E_{ext}(\mathbf{v}))ds, \tag{1.3}$$

where E_{int} represents the internal energy of the spline due to bending and stretching and E_{ext} gives rise to the external constraint forces and image forces.

The internal energy is defined as follows:

$$E_{int}(\mathbf{v}(s)) = \alpha(s)|\mathbf{v}_s|^2 + \beta(s)|\mathbf{v}_{ss}|^2, \tag{1.4}$$

where $\mathbf{v}_s = \partial\mathbf{v}/\partial s$ and $\mathbf{v}_{ss} = \partial^2\mathbf{v}/\partial s^2$. The first-order term $\alpha(s)|\mathbf{v}_s|^2$ makes the snake behave like a string (*i.e.*, it resists stretching), whereas the second-order term $\beta(s)|\mathbf{v}_{ss}|^2$ makes it behave like a rod (*i.e.*, it resists bending). The weight $\alpha(s)$ regulates the tension of the snake, whereas $\beta(s)$ regulates its rigidity.

The external energy, E_{ext}, usually consists of the image potential energy and energies from external constraints. The image potential energy is derived from the gravitational potential energy equation in a point-by-point fashion as follows:

$$E_{ext}(\mathbf{v}(s,t)) = \mu g z(\mathbf{v}(s,t)) \tag{1.5}$$

where μ is the constant mass density of the snake, g is the constant magnitude of the gravitational acceleration, and $z(\mathbf{v}(s,t))$ is the height of snake element s at time t on the surface. External constraints depend on applications and can be qualitative shape descriptions or even intentionally added forces such as *volcanos* in Kass *et al.*'s implementation [31].

To solve the minimization problem, the initial energy equation is converted into Euler-Lagrange equations of motion using the theory of calculus of variations. The Euler-Lagrange equations are as follows:

$$\mu x_{tt} + \gamma x_t - \frac{\partial}{\partial s}(\alpha(s)x_s) + \frac{\partial^2}{\partial s^2}(\beta(s)x_{ss}) = -\frac{1}{2}E_{ext_x}(\mathbf{v}), \tag{1.6}$$

$$\mu y_{tt} + \gamma y_t - \frac{\partial}{\partial s}(\alpha(s)y_s) + \frac{\partial^2}{\partial s^2}(\beta(s)y_{ss}) = -\frac{1}{2}E_{ext_y}(\mathbf{v}), \tag{1.7}$$

where $x_{tt} = \frac{\partial^2 x}{\partial t^2}$, $y_{tt} = \frac{\partial^2 y}{\partial t^2}$, $E_{ext_x} = \frac{\partial}{\partial x}(E_{ext})$, and $E_{ext_y} = \frac{\partial}{\partial y}(E_{ext})$, and γ is the constant damping density or viscosity factor.

The partial differential equations (PDE's) are solved by discretizing them in the two domains of space and time. The discretization must be carefully considered. Different methods exist in the numerical analysis literature for such PDE's. The two most popular methods for solving PDE's are the finite differences method (FDM) [33] and the finite elements method (FEM) [14]. Basically, FEM is a better method but FDM is also adequate for this problem and needs far less computational effort [33]. In the literature for snakes, FDM is the most common choice. The resulting equations are a sparse linear system which can be solved in linear time [33].

As pointed out by Amini *et al.* [3], satisfying the Euler-Lagrange equations is only a *necessary* condition to obtain the minimum of the original functional. The solution using the calculus of variations method could therefore be far from optimal. They instead proposed to use dynamic programming (DP) to solve the minimization problem [2]. The method is, however, computation intensive, but can handle snakes with non-differentiable external constraints. Williams and Shah [54] presented a method using greedy heuristics to search for a solution. It takes less computation effort, but can lack accuracy of solution when there are multiple local minima. Storvik [47] used a Bayesian framework for active contours, and the problem is solved by simulated annealing (SA), which has the ability to avoid local minima.

The original snake was designed as a "power assist" to a person pointing to a contour feature on an image. The initial contour and external constraints are provided by the user. However, there are several problems with this original snake formulation: (1) An initialization, which should be very close to the desired object boundary, needs to be provided by a user. (2) Multiple local minima on an image potential field will prevent a snake from converging to the global minimum. (3) It is quite difficult to choose weights for various internal and external energies.

A number of people have proposed solutions to these problems. Berger [5] proposed a method called *snake growing* to try to eliminate the snake initialization problem. However, his method, in which snakes start from edges selected from an edge map and grow at the two ends, loses the desirable global property of the snake model. It has no global model and thus can only possibly fill small gaps. To cope with the initialization problem, Cohen and Cohen [14] proposed a *balloon* model that uses an inflating force to make the contour expand until it meets an object boundary. However, as pointed out by Xu *et al.* [57], a snake tends to shrink due to its internal forces and, even worse, the internal forces are not homogeneous along the boundary, being large at points with high curvature. This makes it difficult to choose a constant inflating force for the balloon model. Gunn and Nixon [23] presented a dual snake scheme with one snake starting from outside the object boundary and one from inside. The process stops when the two snakes meet each other. Although this scheme greatly reduces the sensitivity of a snake to its initial position, it still needs user initialization.

Another reason that a snake is sensitive to its initialization is because of the existence of multiple local minima. The problem of how to avoid getting stuck in local minima has been investigated in the literature. Choosing a good minimization algorithm is the first issue. In this sense, Storvik's [47] Bayesian model and simulated annealing may be the best choice. The second way to avoid local minima is by using a *coarse-to-fine* multi-scale approach [33]. In this way, the image is first smoothed by a set of Gaussian filters of different sizes, resulting in an *image stack* [55]. The snake is first minimized at the coarsest level, and then this solution is used as the initialization at the next finer resolution, *i.e.* on the next image in the stack. This process is repeated until it reaches the finest resolution. At the coarse levels, details are smoothed out, and thus the snake is readily attracted by prominent features, avoiding local minima. The problem with this framework, however, is that it cannot handle both diffuse and non-diffuse edges simultaneously, both of which are common in noisy images.

The third way to avoid local minima is to pre-process the information available and use multiple information sources. Chiou *et al.* [13] used a neural network-based method to eliminate noise and unnecessary information, resulting in a well-posed potential field. Radeva *et al.* [42] argued that local minima were resulted from the snake's being attracted by edge points that do not correspond with object contours. The method they proposed was to incorporate more information, such as gradient direction and *a priori* shape information, into the snake model. The importance of gradient direction information was also observed by Fua *et al.* [21] and Chakraborty *et al.* [11]. Chalana *et al.* [12] used temporal information by putting several related active contours in a sequence of echocardiographic images. Ivins and Porrill [28] presented active region models (ARMs), which are based on balloons, to incorporate texture and color information. Each time that an ARM expands, the texture or color statistics of the region to be occupied are computed and compared with those of the current region. Expanding is continued if the comparison test is passed according to some predefined test. All of these methods have, to some extent, improved the original snake model.

Several other researchers proposed incorporating *a priori* shape information in snakes to avoid local minima. Olstad *et al.* [39] used a grammatical model to describe the contour. The encodings are used as a hard constraint to the energy minimization process, and the energy can be minimized by a discrete dynamic programming algorithm. Lai [32] proposed a method to describe the boundary shape using a shape matrix. The shape matrix is proven to be scale, rotation, and translation invariant. However, the algorithm used to minimize the energy is also a variant of dynamic programming which is computationally intensive.

As to choosing the regularization parameter λ, only one paper [32] has dealt with this problem. A *minimax* criterion is used to select the regularization parameter λ automatically. The method can provide a solution that avoids extremely low or high weights; however, it requires that a discrete dynamic programming or greedy algorithm be used for energy minimization.

1.2 Why Active Contour Models?

Active Contour Models have since attracted much attention on the part of researchers. In this research, we base our approach on them as well and try to solve some problems associated with them. The major benefit of using ACMs lies in their great ability to combine various information sources such as edges, texture, and shape models. From our analysis, we conclude that ACMs embody many concepts that researchers have identified:

- The initializations required by the original ACMs carry shape information and serve as a global model. Further improvements such as Olstad's grammatical model [39] and Lai's shape matrix [32] confirm that ACMs can effectively use *a priori* information.

- The internal energies that maintain the smoothness of ACMS are a good example of the regularization concept, which is essential to solving ill-posed problems [48]. Regularization techniques, or penalized optimization, are used for many applications in vision.
- An implicit local "voting" process is incorporated for every node on a snake because the neighboring nodes have influences on one another. This concept is best illustrated by the Hough transform.
- ACMs can effectively combine various information sources.

ACMs have been proven to be robust due to their unique ability to combine knowledge, regularization constraints, and various image information sources. Based on this well-formulated framework, our approach inherited the above advantages.

1.3 Active Search and ACMs

The initializations that are needed for ACMs are sometimes viewed as a shortcoming. For example, ACMs cannot be directly used in automatic applications. In that case, a good pre-processing module is required to provide ACMs with accurate initializations.

Lai [32] proposed using the generalized Hough transform to initialize the shape-matrix based snakes in cluttered images. A training set needs to be provided to estimate the shape matrix. The performance of this initializer is not satisfactory when there is severe clutter and deformation. Poor initializations thus yield poor solutions.

Cohen and Cohen [14] instead considered a snake as a balloon that is inflated by a pressure force as if air is introduced inside. The snake then expands while subjected to attracting forces from edges as before, and it will pass over edges which are too small or too weak, thus avoiding being trapped by local minima. Initializations are still needed for balloons, but they are trivial and can be effectively provided by a pre-processing module.

Neuenschwander *et al.* presented the ziplock snake model [38], which needs only two user-supplied endpoints. The optimization process for a ziplock snake starts from the two endpoints and progresses towards the center of the snake. During the process, the image potential is progressively turned on to clamp the two ends of the snake on to an image contour. The ziplock snake model reduces the initialization need for a snake to merely specify two endpoints.

What the balloon models and the ziplock snake models do is different from the original ACMs — they are no longer a final refining tool. Instead, they actively search for the desired contours, as opposed to being passively attracted to image features. In this research, we develop methods that further the practice of *active search*. Our methods are based on balloons and ziplock snakes but are more advanced and reliable. An overview of all the methods is provided in the next section.

1.4 Overview of the Thesis

1.4.1 Objectives of the Thesis

The first objective of this research is to extract contours of objects with fuzzy boundaries. The objects are assumed to have relatively homogeneous intensities or other features. Examples include ultrasound, infrared, and laser speckle images, all of which are being extensively used. The noise pattern, or speckle, in these images, is caused by the interference of energy from randomly distributed scatters, too small to be resolved by the imaging system. Region-based segmentation methods are not effective because the background or neighboring objects have similar characteristics and may be connected via the fuzzy parts. The second objective is to extract contours from images with multiple contours or with heavy clutter. Contours are assumed to be strong but may have missing parts or be tangled with one another. This category of images includes most images we see everyday.

1.4.2 Contributions of the Thesis

Active search is the basic concept of our approach, which is based on two variants of ACMs: balloon models and ziplock snake models. These two kinds of models either expand or grow to search for contours in a large area. Our method, dubbed ASCMs, thus needs only trivial initializations and can be used in automatic applications. Fig. 1.1 shows the relationship between ASCMs and the segmentation methods in the literature.

During the searching process, ASCMs combine information from various sources:

- Raw image information, which includes edge and region features, is used effectively in ASCMs.
- Mid-level information produced by a special pre-processing method for ultrasound-like images. The pre-processing method uses local information to reduce noise, and therefore it is also useful for non-ultrasound images.
- Information collected during the searching process is used to predict and/or reinforce searching directions.
- Shape-models, which possess high-level information, are adopted to direct the searching process and to provide an informed guess.

The last three items in the above list are our contributions.

The balloons and ziplock snakes are primitive in terms of their searching power. Also, as described in Section 1.1, it is very difficult to choose an appropriate inflating force for balloons. In this research, we present a new search engine which is more powerful and controllable. Features of this new searching module are shown in the following list, which also serves to summarize the contributions of this thesis.

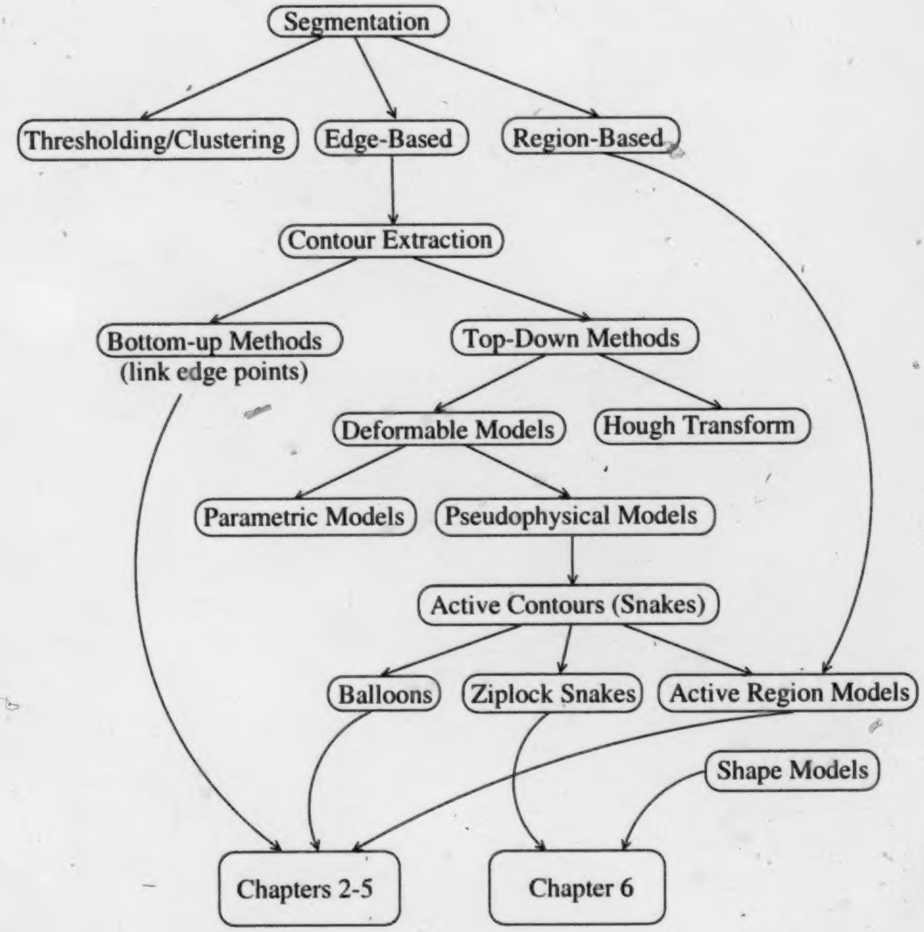

Segmentation
Thresholding/Clustering
Edge-Based
Region-Based
Contour Extraction
Bottom-up Methods
(link edge points)
Top-Down Methods
Deformable Models
Hough Transform
Parametric Models
Pseudophysical Models
Active Contours (Snakes)
Balloons
Ziplock Snakes
Active Region Models
Shape Models
Chapters 2-5
Chapter 6

Figure 1.1: The relationship between ASCMs and the segmentation methods in the literature.

- To help reduce the difficulty of choosing an inflating force, we propose a non-shrinking internal energy model that is scale, rotation, and translation invariant.
- To provide smoothing power at various scale levels, we propose a multi-scale energy model which achieves its objective through the use of multiple snakes.
- To reduce the side-effects of using a constant inflating force, we propose using an adaptive inflating force which embodies the idea of regularization. The adaptive inflating force is computed for every snake node based on a small neighborhood around it. The computed force is "just enough" to make a snake expand and is set to be trivial in places where image features are not present. This leaves the smoothing forces in a dominant position, which amounts to providing a regularized result.
- To perform a multi-level feature search, we propose incorporating a backtracking process, which can not only overcome local minima but also effectively use both strong and weak image information.
- To better approximate a spline, we propose a semi-rigid model which models joints of a discretized snake as spring-operated hinges. This new model is closer to the definition of splines than the original string and rod models.
- To actively search for contours, we present a content-guided searching method that uses outputs of an edge linking-like method to guide the searching process. This method is a combination of a bottom-up and a top-down method. The method not only enables a snake to reach high-curvature boundary parts but also expedites the searching process.

The thesis is divided into two parts (see Fig. 1.2). The first, which includes Chapters 2-5, is based on the balloon models and is designed to achieve the first objective, *i.e.* to extract contours of objects with fuzzy boundaries. For a balloon, the essential properties it needs to be successful are: (1) an ability to perform contour completion; (2) to be able to overcome local minima; and (3) to provide ease of parameter choosing. Properties (1) and (2) are not only difficult to achieve but also contradictory to each other. For the original balloon model, only a compromise can be achieved through careful adjustment of parameters. In this research, we delegate the desired functions to several separate modules. A global strategy, which is based on image content, time, and current results, is then used to control these modules during the energy minimization process. In this way, objectives (1) and (2) can be achieved at the same time. Parameter adjustment is made easy because parameters are now separated from one another.

The second part (Chapter 6) is based on Neuenschwander *et al.*'s ziplock snake models and is aimed at extracting contours from a cluttered background. The original ziplock snakes are easily confused by multiple contours, missing contour parts, or noise. To overcome these problems, a grammatical shape model is added to guide the searching process of the ziplock snakes. The shape information is encoded using

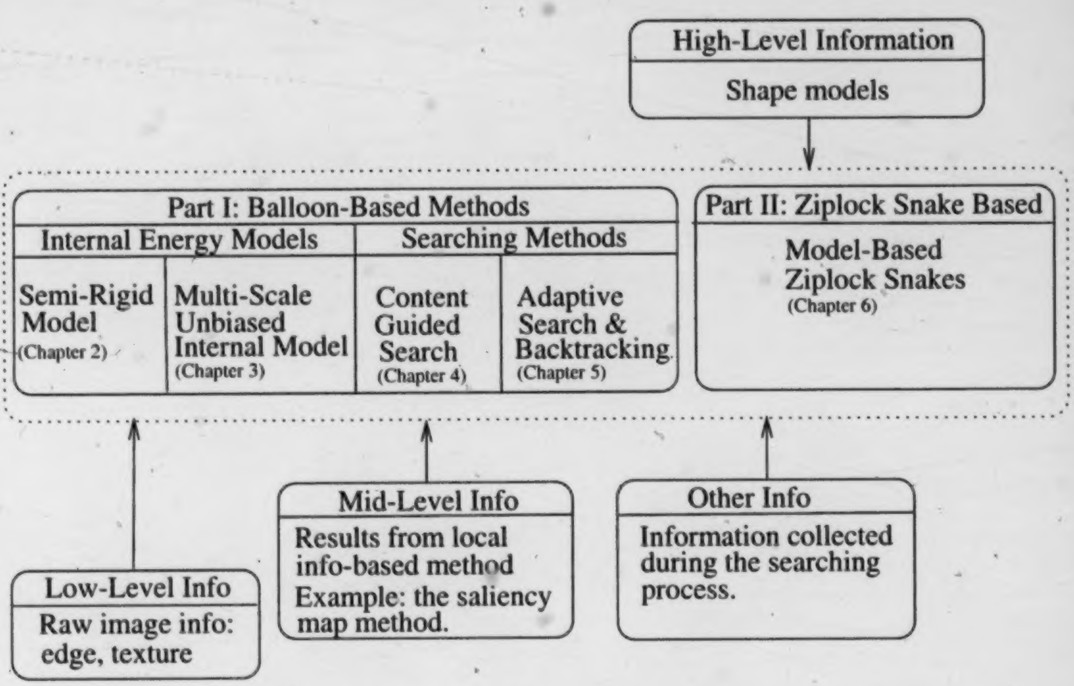

High-Level Information
Shape models
Part I: Balloon-Based Methods
Internal Energy Models
Searching Methods
Semi-Rigid Model (Chapter 2)
Multi-Scale Unbiased Internal Model (Chapter 3)
Content Guided Search (Chapter 4)
Adaptive Search & Backtracking (Chapter 5)
Part II: Ziplock Snake Based
Model-Based Ziplock Snakes (Chapter 6)
Low-Level Info
Raw image info: edge, texture
Mid-Level Info
Results from local info-based method
Example: the saliency map method.
Other Info
Information collected during the searching process.

Figure 1.2: An overview of ASCMs: components that comprise the whole system.

attributed regular expressions, which are easy to understand and implement. The resulting model-guided ziplock snakes are able to tell model-conforming contours from those that are not, and, thus, their performance is much better. The final section of the thesis, Chapter 7, provides a list of contributions and a brief discussion of future work.

Chapter 2

Semi-Rigid Snake Models

2.1 Introduction: The Peeling Problem

The balloons were created by incorporating an inflating force into the original ACMs. However, this practice also gives rise to some side effects. We observed a phenomenon that we call a "peeling" effect in applying the balloon models to extracting contours with missing or weak parts from noisy images. To overcome local minima, we have to choose an inflating force large enough to overcome the noise edges. However, this will also cause some parts of the balloon to overrun some weak edges. Then these parts will pull their neighboring contour parts away from locations that are otherwise in equilibrium.

To understand the peeling problem, some illustrations are provided. Fig. 2.1 (a) shows a synthetic boundary with one gap on it. The boundary magnitude changes gradually from small to normal from the gap. If we choose an inflating force that can overcome an average magnitude edge, a probable intermediate state of the balloon is shown in Fig. 2.1 (b). Now we check the status of the two neighboring snaxels[1], x and y, with x on the gap and y on the brink of the gap. Each of them is under the influence of three forces, namely the gradient force, the internal smoothing force, and the inflating force. We assume that snaxel y is now in equilibrium under the influence of these three forces. For x, the gradient force is small compared with the inflating force, so it will continue to move outward (Fig. 2.1 (c)). This will cause a change in the internal force exerted on y since the internal angle θ is getting bigger. Eventually, this force will pull y away from its equilibrium state, and it will reach a state as shown in Fig. 2.1 (d). This process will continue with y's upper neighboring snaxels and eventually produce a large bump. This is what we call the "peeling" problem.

The reason that the peeling problem occurs lies in the way that the internal energies are computed and that the total energy is minimized. We argue that using second-order internal energy models such as those that are curvature-based and

---

[1] After discretization, a snake consists of a series of snake nodes called *snaxels*, which is an abbreviation of snake element. In Chapters 2-5, all snakes are closed and are in anti-clockwise order, which means that each snaxel has a predecessor and a successor.

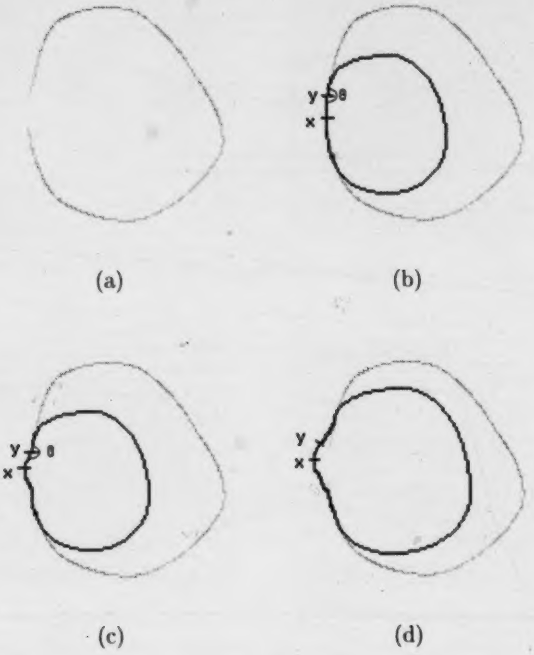


Figure 2.1: An illustration of the "peeling" problem: (a) a synthetic boundary; (b) a probable intermediate state of the balloon; (c) snaxel x keeps moving outward; d) snaxel y is pulled away from its equilibrium location.

a small neighborhood in energy minimization is what causes the peeling problem. Fig. 2.2 shows three states of a balloon, which is a partially magnified version of the one shown in Fig. 2.1. Assuming again that the upper half of the balloon is snagged by edges and the lower half is going through a gap, the only difference in total energy among the three states is the internal energy computed at snaxel y. This energy difference is proportional to the difference between the internal angles (θ's). To make the balloon expand from the state (a) to (b), and then to (c), the inflating force need only defeat the internal energy difference at snaxel y. Moreover, since the total energy is minimized iteratively, each time only a small displacement for snaxel x is considered (Fig. 2.2 (d)). A small displacement involves a small angle difference, which in turns produces a small internal energy difference. Therefore, a small inflating force is enough to defeat this small internal energy difference. Although the internal angle θ at snaxel y increases steadily, it has no effect on other snaxels' internal energy computation. Snaxel y will eventually move again to dissipate the accumulated energy. The process will repeat on y's immediate upper neighbor and so on.

The ultimate reason for this behavior is that a snake is modeled as a series of nodes connected by strings. For snaxel y, the internal energy increases constantly throughout the iterations. However, this energy only produces a force to make y

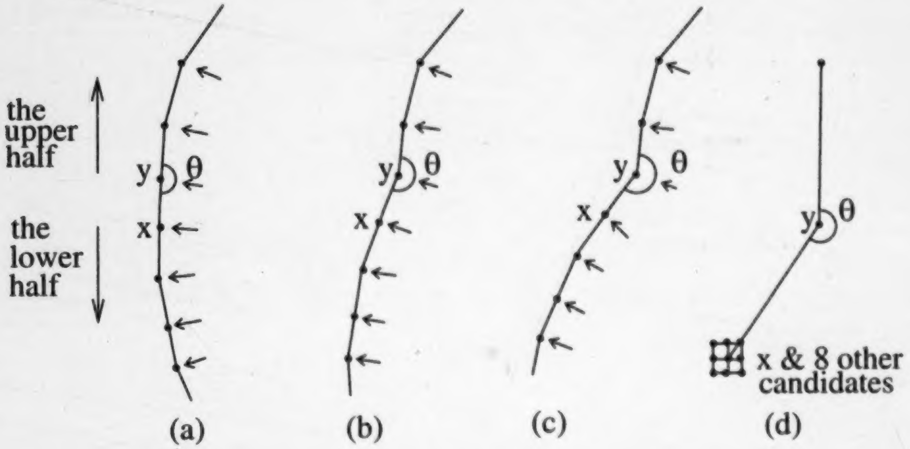


Figure 2.2: How the peeling problem is related to the internal energy computation method: (a-c) three consecutive states of a snake; (d) for snaxel x, a small neighborhood (a 3x3 grid) is considered when its energy is minimized.

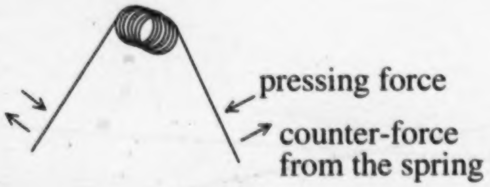


Figure 2.3: A spring-operated hinge is a more accurate model of a spline.

move. Ideally, it should also produce a counter-force to prevent x from moving further ahead so as not to increase the internal angle θ. However, strings cannot produce this kind of force because they are soft links, and the internal energy computation only involves a snaxel's two immediate neighbors.

A better model will be a series of short rods jointed together with spring-operated hinges (*e.g.* see Fig. 2.3). A spring-operated hinge produces large counter-forces when it is pressed hard. In this way, gradual internal energy build-up processes will be avoided, thus reducing the chances of the peeling problem occurring.

A simulation is performed to show the advantages of rod-based models. In Fig. 2.4, a partial balloon with 4 fixed snaxels (y_{1-4}) and 3 moving snaxels (x_{1-3}) is instrumented to record internal energies for every snaxel during energy minimization. The dotted line indicates the starting positions of all snaxels. Fig. 2.5 plots the internal energy of snaxel y_2 with and without y_2 and y_3 modeled as spring-operated hinges. We observe that the internal energy of y_2 is reduced when y_2 and y_3 are modeled as hinges, and thus they are less prone to the peeling problem.

In this chapter, we propose a semi-rigid snake model that aims at preventing the peeling problem. An iterative process is first presented to dynamically identify snake

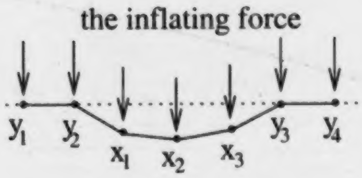

the inflating force
y1
y2
x1
x2
x3
y3
y4

Figure 2.4: A simulation configuration: snaxels y_{1-4} are not movable while x_{1-3}'s positions are determined by the internal energy and the inflating force.

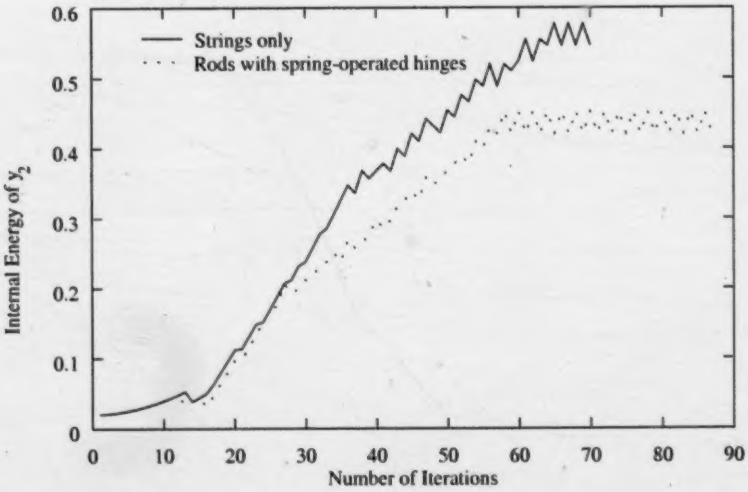


Figure 2.5: The internal energy of y_2 during iteration modeled as a string or a rod with spring-operated hinges.

parts that have successfully located strong boundaries. Once a few parts have been identified, an influencing force is propagated from these parts to their neighbors to counter the peeling force. This influencing force is similar to that generated by a spring-operated hinge, and it makes a snake act like a semi-rigid object. To prevent side-effects, only the snake parts that are on top of strong edges (called strong snake parts) are made semi-rigid. The identification of the strong snake parts and the following *"rigidization"* process are dynamic and are integrated into the minimization process.

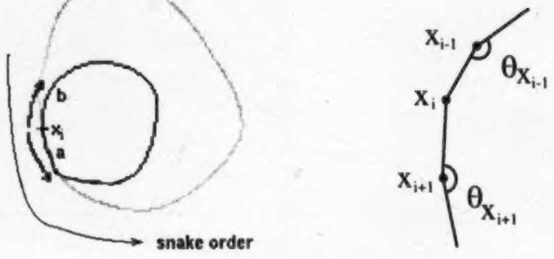
2.2 Dynamic Strong Snake Parts Identification

Modeling a snake as a series of rods connected with spring-operated hinges takes it one step closer to the definition of a spline. Computation is quite complicated if the positions of all hinges are constantly changing. To reduce the computation, we propose a simplified model. First, a procedure for identifying strong snake parts is developed. Then each of the strong snake parts is modeled as a large combined hinge, and influencing forces are generated for its neighbors at both ends. The differences between the complete and this simplified model are two-fold: (1) A snake is now modeled as a combination of strings and rods, *i.e.*, only the strong snake parts are modeled as rods with spring-operated hinges, thus reducing computation. (2) The influencing force computation for this simplified model is asymmetrical, *i.e.*, it considers influences only from strong snake parts to weak ones, not vice versa. This is a reasonable assumption because it reflects the way that the Human Vision System (HVS) operates.

We now try to devise a method to dynamically identify strong snake parts. For each snaxel, an appropriate measure is defined and computed. Naturally, the measure for a snaxel should be large if it is on top of a strong boundary. The task is similar to identifying salient structures in images [44, 24], except for two differences. First, the latter task involves a two dimensional search, *i.e.*, for each pixel, every direction should be checked for possible connections with salient structures. However, for a snaxel, we only need to check two directions, namely, its predecessor and successor on a snake (Again, snakes are in anti-clockwise order). Second, a snake is always evolving, so the computation should be dynamic as well. Nonetheless, the guidelines presented in these two papers can still be used.

Since our final objective is to use the measures of some snaxels to direct the movement of their neighbors (*i.e. rigidization*), we represent them in terms of influence factors. Since the influence factors are bi-directional, two measures should be defined for each snaxel. For a snaxel x_i, let $I_p^{x_i}$ stand for the influence on x_i's predecessor computed from its successor's side of the snake and $I_s^{x_i}$ the opposite one. In Fig. 2.6 (a), $I_p^{x_i}$ should be larger than $I_s^{x_i}$ assuming that the lower part of the snake, a, lies on a strong boundary, while the upper part, b, does not. Some qualitative guidelines for computing these two measures are listed as follows:

- Strong support from image features — Large gradients are often used for this



(a) (b)

Figure 2.6: An illustration for influence-measure computation: (a) influence measures for snaxel x_i are based on x_i's neighboring parts on a snake, namely a and b. (b) angles used in Eqns 2.1 and 2.2

purpose.

- Strong connections — Since snaxels are separated, image features along the connecting route between them are an important contributing factor to the measures.
- Smoothness constraint — Smooth curves are preferred over non-smooth ones.

A simple way to compute these measures is to calculate an integral of image features along with other factors for a fixed length l starting from each snaxel (Fig. 2.6 (a), a and b for snaxel x_i). However, the complexity of the algorithm is $O(n*l)$ where n is the number of snaxels. We propose using a simpler algorithm with a complexity $O(n)$. The algorithm is used in an iterative way so that influence measures are propagated along the snake, thus having the same effect as computing an integral.

The algorithm can be easily integrated into either the greedy heuristic energy minimization method proposed by Williams *et al.* [54] or the time-delayed dynamic programming method suggested by Amini *et al.* [3]. After each round of computation, I_p and I_s are updated as well (see the pseudo codes in Fig. 2.7). Starting from the head of the snake, I_p is updated for every snaxel in clockwise order using Eqn. 2.1. I_s is also updated in a similar way using Eqn. 2.2 except that the snaxels are traversed in the opposite order:

$$I_p^{x_i} = I_{image} + I_p^{x_{i+1}} C(x_i, x_{i+1}) \cos(\theta_{x_{i+1}} - \theta_{avg}); \tag{2.1}$$

$$I_s^{x_i} = I_{image} + I_s^{x_{i-1}} C(x_i, x_{i-1}) \cos(\theta_{x_{i-1}} - \theta_{avg}), \tag{2.2}$$

where I_{image} is the normalized magnitude of gradient at snaxel x_i. $C(x_i, x_{i+1})$ is a measure of the connection between the current snaxel, x_i, and its successor, x_{i+1},

```
procedure MinimizeEnergy()
begin
    repeat
        update energy for every snaxel;
        move all snaxels to their next position;
        update $I_p$ for every snaxel in clockwise order;
        update $I_s$ for every snaxel in anti-clockwise order;
        compute influencing forces for every snaxel;
    until (Terminal Conditions = true);
end
```

Figure 2.7: Pseudo codes for computing influence measures.

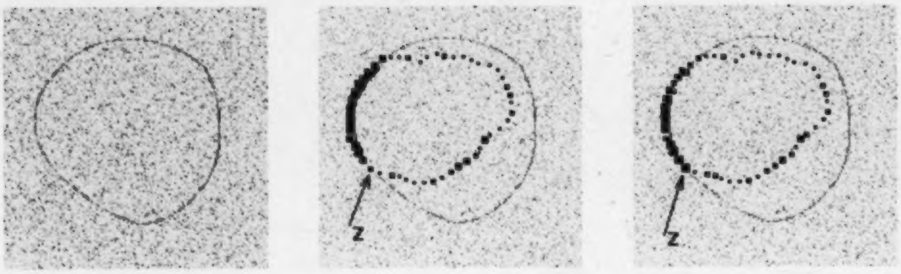
and is computed by mapping the average gradient magnitude along the straight line that connects the two snaxels through an exponential function, $1 - exp(-\rho t)$, where t is the normalized value of the average gradient magnitude. $C(x_i, x_{i-1})$ is computed similarly. Finally, $\theta_{x_{i+1}}$ is the internal angle formed by the current snaxel and its two successors (Fig. 2.6 (b)), $\theta_{x_{i-1}}$ is the internal angle formed by its two predecessors, and θ_{avg} is the average internal angle.

Eqns. 2.1 and 2.2 embody the guidelines that we presented above. The function $C(x_i, -)$ introduces the connections between the current snaxel and its two immediate neighbors. $\cos(\theta - \theta_{avg})$ is designed to favor influences from smooth neighboring snaxels. I_{image} is a contribution from the snaxel in question. Through an iterative update of the measures, influences from one snaxel can propagate to distant snaxels depending on how parameters are chosen. From the example shown in Fig. 2.8, we observe that for almost all snaxels that are inside the boundary, both I_p and I_s are small. Moreover, the few with relatively large measures are not able to propagate this information to their neighbors because of poor connections between themselves and their neighbors. For snaxels that lie on a piece of strong boundary, there are some noticeable differences between I_p and I_s. Snaxel z indicated by arrows in Figs. 2.8 (b) and (c)) is a good example of a case in which I_p is smaller than I_s as expected.

2.3 Computing Influencing Forces

Once the strong snake parts have been identified, they are converted from string-linked nodes into hinge-jointed rods. This process is called "rigidization" because the snake is converted into a semi-rigid object. From an alternative perspective, identifying strong snake parts is the same as collecting information on what a snake has found. This information is useful and can be used to direct the search of the neighboring parts of those strong snake parts. This is why the method is also called the extrapolating mechanism.

To be consistent with snake models, "directing" is applied by adding one more kind of stiffness force. The forces are generated from strong snake parts and are



(a) (b) (c)

Figure 2.8: An example of influence measures: (a) a synthetic boundary with noise; (b) an example of I_p in which the size of the snaxels indicates the magnitude; (c) the corresponding I_s.

applied to their neighboring parts. There are two problems that should be solved to make the algorithm work:

- How to judge whether a snake part is indeed a strong one. How large should the influence measures be to qualify as a strong snake part?
- How to compute the directions and how to assign forces.

Our current solutions are relatively simple and work with a discretized snake directly. For each snaxel, the influence measures from its predecessor and successor are compared. If one is larger than the other by a certain amount δ, then the side that the bigger influence is from is considered a part of a strong snake part; influencing forces are generated and applied to the other side. By using relative measures instead of absolute ones, we circumvent the problem of choosing a threshold for strong boundaries. Experiments show that choosing a difference threshold for two measures is relatively easy and that one value works with most cases.

To generate influencing forces, we need to compute the direction and magnitude of the forces. Fig. 2.9 is an example of how we do so. Assuming that snaxel x is the one in question and y is its successor with a large influence measure, our task is to use the shape, distance, *etc.*, of the part a to predict a position for x. A natural way to do so is to extend the curve along the tangent (Fig. 2.9 (a)); or we can ensure that the internal angle shown in Fig. 2.9 (b) as θ equals to the average internal angle while extending the curve. The differences between the two methods are small. We chose to implement the latter and use the positions of the two immediate snaxels.

Once the curve-extending method is decided, the predicted position for the current snaxel is calculated simply by extending the curve by a length of l, which is the distance between snaxels x and y. In Fig. 2.9 (c), positions predicted from both sides of x are shown as z and z'. Choosing which one to use depends on which of the measures is bigger. If $\mathbf{v}$ is the influencing-force vector, we set

$$\mathbf{v} = z - x. \tag{2.3}$$

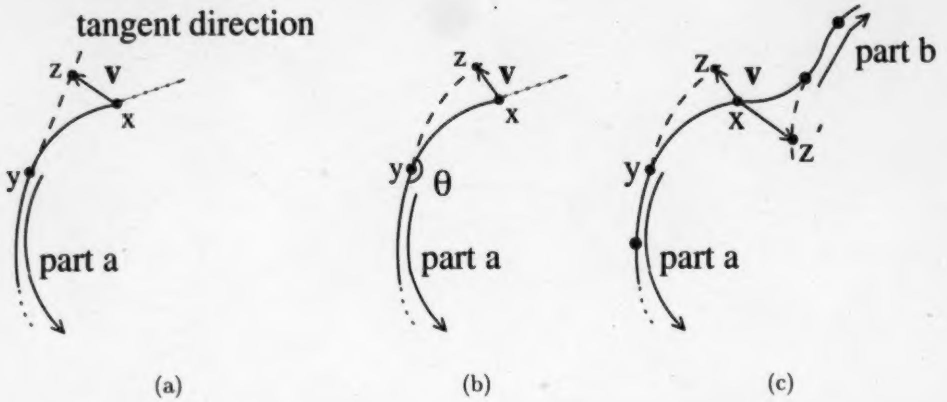


Figure 2.9: An example of generating influencing forces: (a) part a is extended in the tangent direction; (b) extended by maintaining the internal angle; (c) curve-extending is done from both directions.

2.4 Implementation and Experiments

We evaluate our method by comparing results generated by a balloon model equipped both with and without the extrapolating mechanism. Testing images include some synthetic line-drawings with different noise levels and some real noisy images.

First, some implementation details are introduced. We implemented the time-delayed dynamic programming method [3] for both balloon models (See Appendix A for a brief account of the method). Each snake starts with a rough initialization inside the boundary that we are looking for. Additional snaxels are added at places where the distance between consecutive snaxels exceeds a predefined threshold. In the experiments shown below, the distance between snaxels is maintained at 5 pixels.

The balloon forces and the influencing forces produced by the extrapolating mechanism are converted into a potential field so that they can be used along with other energy functions. Assuming that $\mathbf{v}$ is the influencing-force vector for snaxel x (Fig. 2.10), the potential energies for each of $x's$ 8 neighbors are assigned in the following way:

$$E_{influ} = -\frac{1}{2}\kappa[\bar{\mathbf{v}} \cdot \bar{\mathbf{w}}] \left|\frac{\mathbf{v}}{h}\right|^2, \tag{2.4}$$

where κ is a scaling factor. Barred symbols are unit vectors, and $\bar{\mathbf{w}}$'s direction is from x to each of its neighbors. The operator $\cdot$ is the usual vector dot-product. The constant h is the average length between snaxels.

A snake may oscillate under the influence of all these forces [33]. A simple termination criterion, such as to see whether any snaxel is moving, is usually unsuccessful, so we devised a more robust method based on a list of the 5 latest history positions

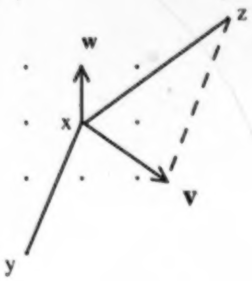


Figure 2.10: Converting forces into potential energies (Eqn. 2.4).

Parameter	ρ	κ	δ	inflating force
Value	1.5	2.0	0.5	0.35

Table 2.1: Parameters used in the following examples unless otherwise specified.

recorded for each snaxel. Whenever a snaxel is moved, its new position is entered into the list and compared. If it is at least two pixels away from any of the history positions, it is considered active. After all snaxels are considered not active for 2 or 3 rounds, the current snake state is considered the final solution. This termination criterion is used to check whether a specific snaxel is oscillating among two or three positions. Since a snaxel can only move one pixel away from its current position in one step, checking whether it is two pixels away is a reliable criterion to use. The number of history positions can be lowered to 3, and the criterion is still quite reliable. While this criterion is ad hoc, similar approaches have also been used successfully [54].

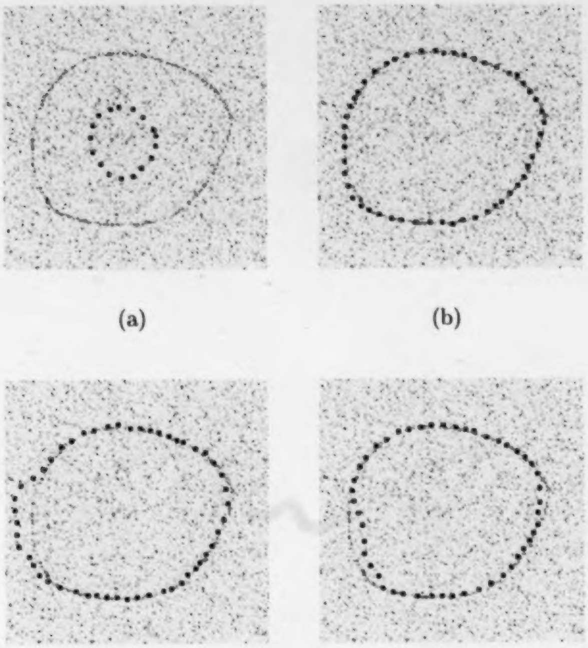
As mentioned before, choosing parameters is a difficult job for a balloon model. In our experiments, the best set of parameters is chosen by trial and error and is listed in Table 2.1. Some good guidelines for choosing parameters can be found in [33, 14]. The scaling factor for the influencing force κ is chosen to be of the same order as those for other forces, such as the balloon force.

In this thesis, we use noisy line-drawings instead of images with shaded regions as test images. Although synthetic images with shaded regions are a better approximation of real life images, we can always derive line images from shaded region images using an edge or line detector. These two categories of images are equivalent for our testing purposes because we do not use region information.

The first illustration uses a synthetic line-drawing image (Fig. 2.11 (a)). Due to the high level of noise, using a constant balloon force produces unsatisfactory results. Two typical results are shown in Figs. 2.11 (c) and (d), while the result produced by our method is shown in Fig. 2.11 (b). Aided by the extra stiffness force from the extrapolating mechanism, we can choose a larger balloon force to deal with noise edges without fear of pushing the snake out of range.

Fig. 2.12 illustrates how the new model copes with big gaps on boundaries.

Fig. 2.13 shows an example that is used to test the robustness of the model against bad initializations. One thing to note is that, in this case, the stiffness forces need to



(a) (b)

(c) (d)

Figure 2.11: An example using a synthetic line-drawing image: (a) the test image with an initialization; (b) the result produced by the semi-rigid balloon models with a normal force of 0.35; (c) and (d) are the results produced by the original balloons with normal forces set to 0.35 and 0.32 respectively.

be larger than the normal force.

The next experiment uses a cup image with fuzzy boundaries. Figs. 2.14 (a) and (b) show the image with an initialization and our solution respectively. Fig. 2.14 (c) is an intermediate result of the original balloon model. Note how the peeling problem takes place and results in a worse result (Fig. 2.14 (d)).

To systematically test the ability of the semi-rigid snake models, we provide a more accurate testing procedure. First, a series of synthetic images with a *unit* image contrast are generated. Each test image contains a circle (with a radius of 50) with four gaps ranging in size from 0 to 60. Zero-mean Gaussian noise with standard deviations from 0 to 1.0 is then added to each image. An example is shown in Fig. 2.15.

To assess the performance, we define several performance measures: e_{FD}, MSE_{nongap}, and MSE_{whole}. The first measure, e_{FD}, is used to assess the smoothness of a snake by computing the feature distance between a snake's Fourier Descriptors [41] and those

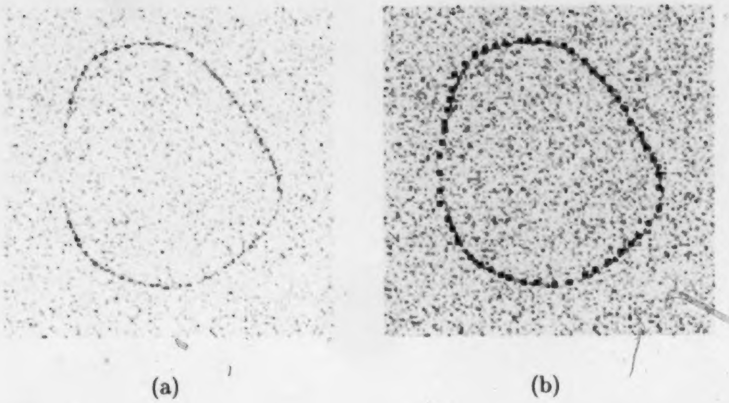

(a) (b)

Figure 2.12: The semi-rigid balloons can cope with gaps on boundaries: (a) the test image; (b) the final result.

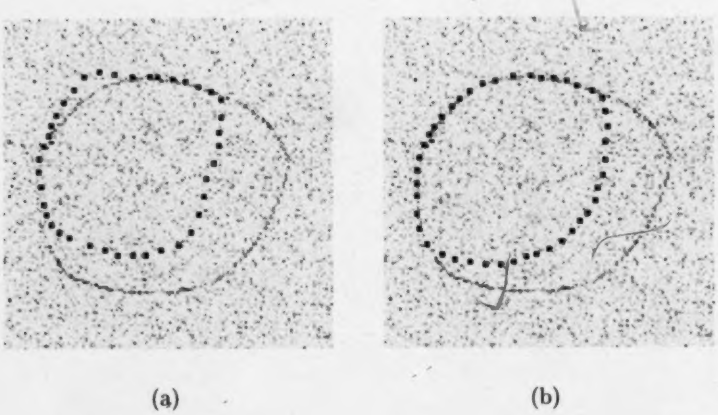
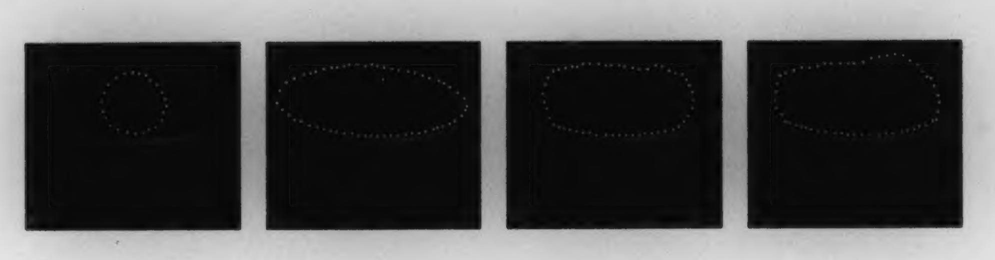
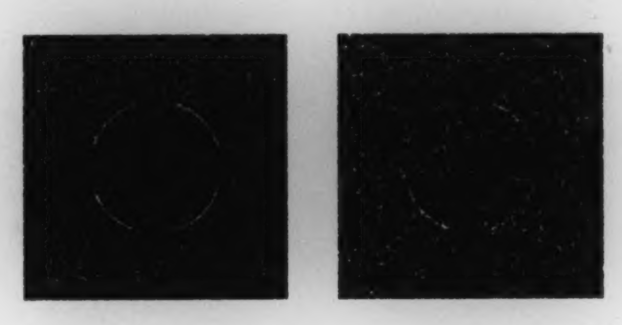

(a) (b)

Figure 2.13: For some bad initializations (e.g. (a)), the semi-rigid balloons can recover from the error ((b), the snake after a few iterations).



(a) (b) (c) (d)

Figure 2.14: A cup image example: (a) an initialization; (b) a result using the semi-rigid models; (c) & (d) results using the original balloons.



(a) (b)

Figure 2.15: Two test-image examples with a noise level of 0.40 and 0.80, respectively. The circle has a radius of 50 and four gaps 30 pixels in length.

of a discretized circle with the same length:

$$e_{FD} = [\sum_{n=-M}^{M} |u_n - v_n|^2]^{\frac{1}{2}},$$

where u and v are the Fourier Descriptors of the two contours respectively.

The last two measures compute the mean squared error (MSE) between the ground-truth circle and a balloon's final shape. The two contours are first aligned by finding the minimal MSE between them. MSE$_{whole}$ is then set to the minimal MSE, and MSE$_{nongap}$ is computed by omitting those snaxels that are on gaps:

$$MSE_{whole}^2 = \min_s E[|d(p(l), \hat{p}(s+l))|^2],$$

where $p(l)$ is the point at l on the parameterized ground-truth circle and $\hat{p}(s+l)$ is the corresponding point on the balloon.

When driven by a normal force, a balloon may never reach a stable state. In the following tests, a balloon is stopped when its length exceeds 130% of the length of the ground-truth circle. The above performance measures are still computed, but their use is limited. In the following plots, the non-stable final results are marked with squares.

The first experiment is designed to show when the peeling effect occurs. Two images, one with a gap size of 30 and a noise level of 0.0, the other with the same gap size but with the noise level increased to 0.40, are used. Both the original balloon models and the semi-rigid models are tested using these two images several times with the normal force increased every time. The performance measures, MSE_{nongap} and MSE_{whole}, are plotted against the normal force in Figs. 2.16 and 2.17. From these two figures, we can see that the semi-rigid balloons are less affected by the peeling problem. When the noise level is increased, the peeling problem makes it difficult to choose an appropriate normal force (see Fig. 2.17). Only the normal force 0.3 enables the original balloon to successfully extract the circle. With the rigidization process enabled, choosing a normal force is less difficult.

The second experiment tests the effects of the scaling factor κ against a changing normal force. The results are plotted in Fig. 2.18. We observe that, for a certain noise level, the normal force has to be larger than a certain threshold (0.30 in this case) for the balloon to perform well. A small κ does not perform well when the normal force increases because there is not enough stiffness energy to overcome the peeling problem. On the other hand, too large a κ may not perform well either. This is because too much stiffness energy may compromise the effects of other energies, resulting in worse results. Generally, the extra stiffness energy should be adjusted in the same order as the other energies.

The third experiment tests the effect of the parameter ρ used in computing the influence measures (Fig. 2.19). As in the last experiment, the semi-rigid models perform best with a medium ρ $(1.5 - 2.0)$. The models also perform better with ρ set to 1.5 than to 2.0 when the normal force is small. Again, we conclude that ρ should be adjusted to match the normal force.

The next experiment compares the performance of the semi-rigid models and that of the original balloons under various noise levels. From Fig. 2.20, we can see that the semi-rigid models perform much better than the original balloons. Even when both of them cannot reach a stable solution, the semi-rigid models have much smaller errors.

The last experiment tests the effects of the influence-measure difference threshold δ (Fig. 2.21). A small threshold introduces too much stiffness energy from noise, thus preventing a balloon from expanding. When the threshold is made large enough (> 0.4), the semi-rigid models perform well consistently.

2.5 Conclusions

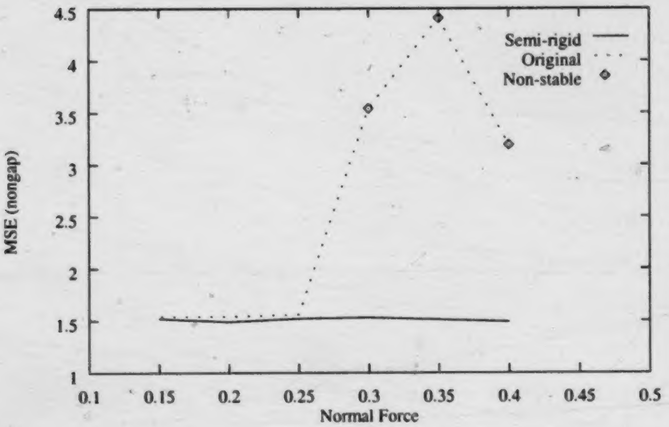
The semi-rigid balloon models are presented to cope with the peeling problem we observed in applying the balloon models to noisy images. The peeling problem is caused by modeling a balloon as a series of string-connected nodes. We instead model some of the balloon nodes as spring-operated hinges, a way that can better approximate a spline.

A procedure that identifies strong snake parts is first presented. The procedure is designed to be iterative; measures are computed by accumulating influences from distant snaxels and, thus, are quite reliable. Those parts identified as strong ones are then converted into semi-rigid objects, a process called rigidization.

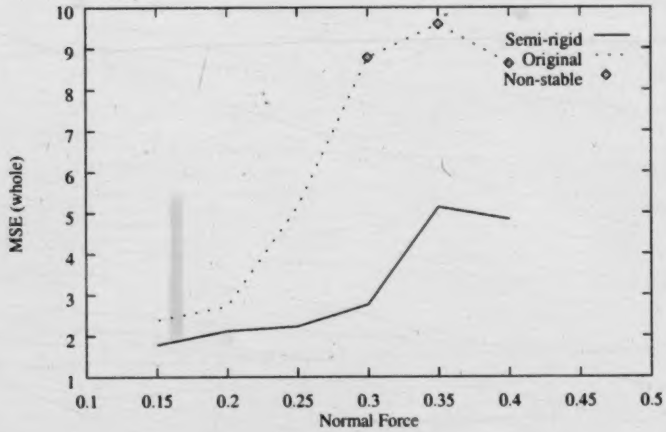
From another perspective, the semi-rigid models collect information while evolving. The collected information is then used to direct the searching of neighboring snaxels. This reflects the way that humans extract boundaries: first we locate the obvious boundaries, and then we use the shape and position of these parts to help locate other parts.

A series of experiments is conducted to show the effectiveness of the semi-rigid models. We conclude that the rigidization process is essential to balloon models because without it balloons often cannot even reach a stable state. Choosing a right normal force is painful for the original balloons. With the rigidization process, the range of effective normal forces is greatly increased.

Finally, since there is no *a priori* shape information involved, the semi-rigid balloon models are not able to recover noise-immersed corners. In such a case, a second-step model-based method should be used to refine the results (see Chapter 4).

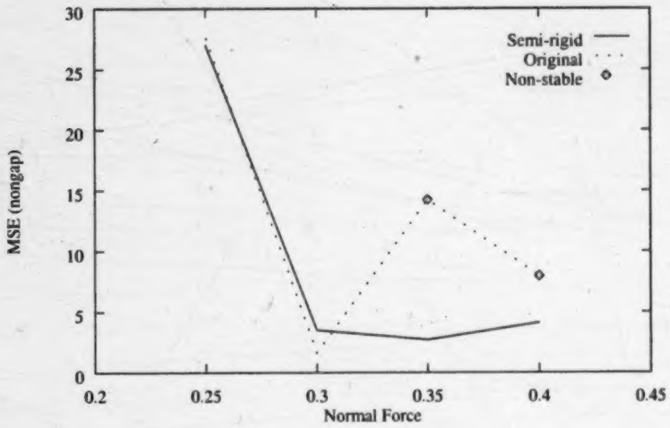


(a)

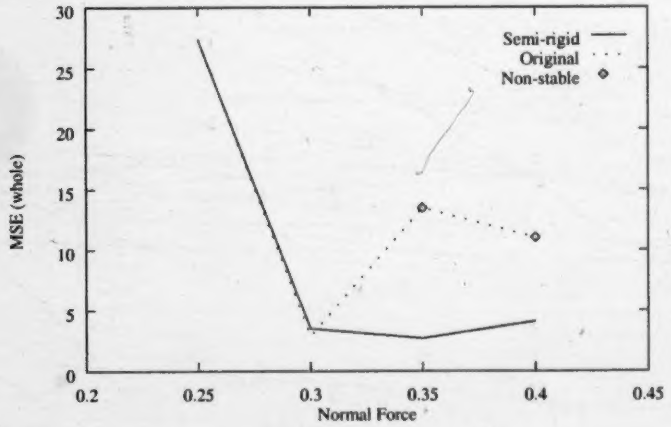


(b)

Figure 2.16: A test on when the peeling effect occurs. The test image has a gap size of 30 and a noise level of 0.0. The semi-rigid models perform consistently better than the original balloons and can reach a stable state even with a large normal force.

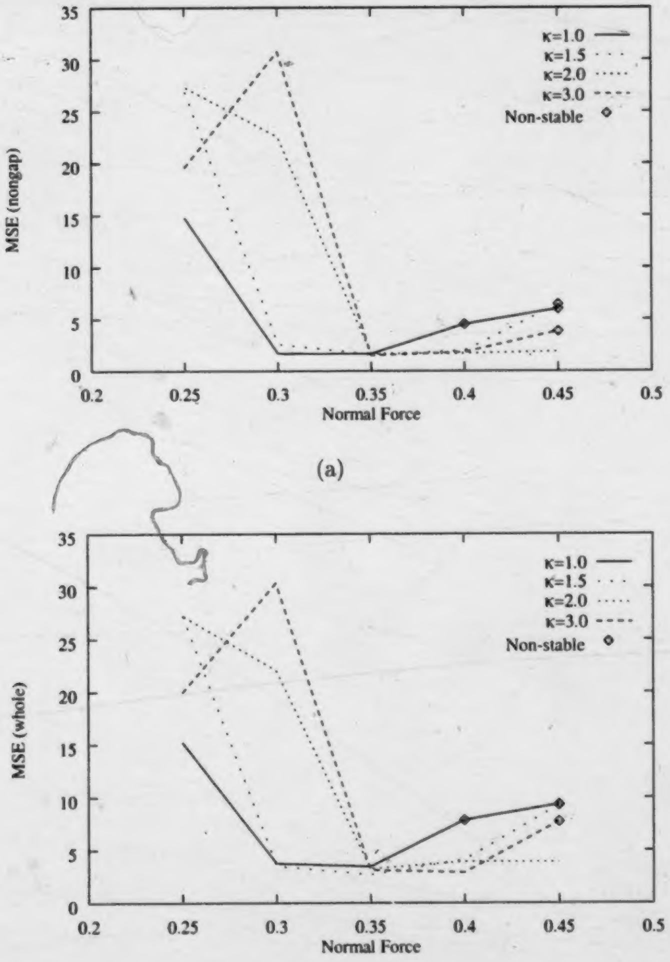


(a)



(b)

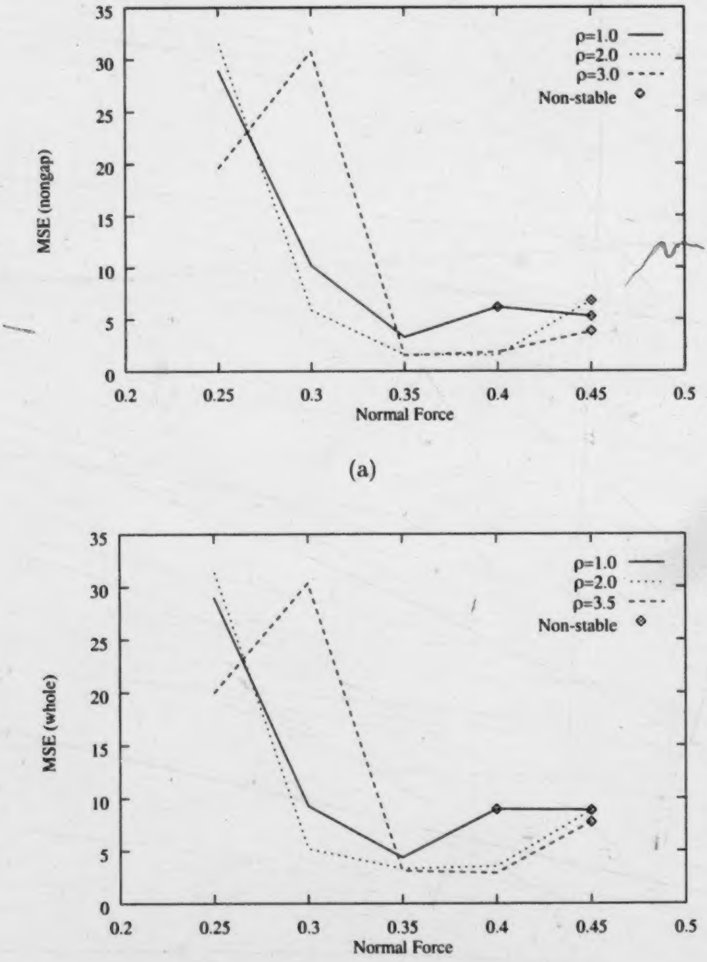
Figure 2.17: Another test on when the peeling effect occurs. Fig. 2.17 shows the same test image except that the noise level is now increased to 0.40. With the noise level increased, the normal force has to be larger than a certain threshold (0.3) for the balloons to perform well.



(a)

(b)

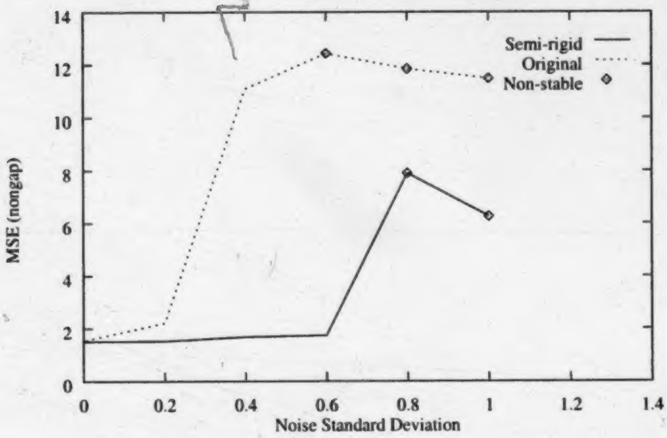
Figure 2.18: Effects of the force scaling factor κ. The test image has a gap size of 30 and a noise level of 0.4. Once the normal force is larger than a certain threshold (0.30 in this case, related to the noise level), the semi-rigid models perform best with a medium κ (2.0).



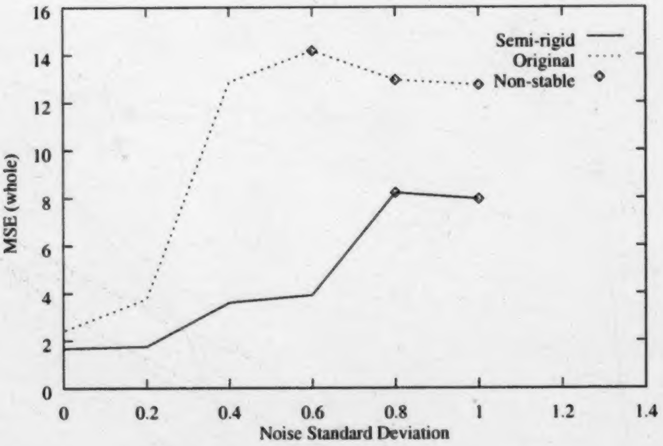
(a)

(b)

Figure 2.19: Effects of the parameter ρ. Gap size = 30, and noise level = 0.4. The models perform best with a medium value (1.5 – 2.0).

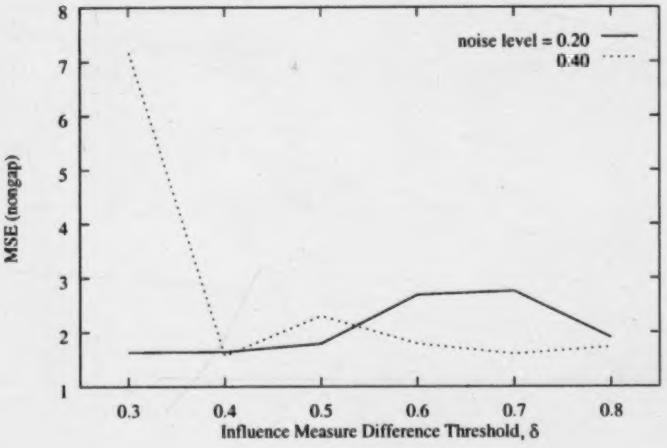


(a)

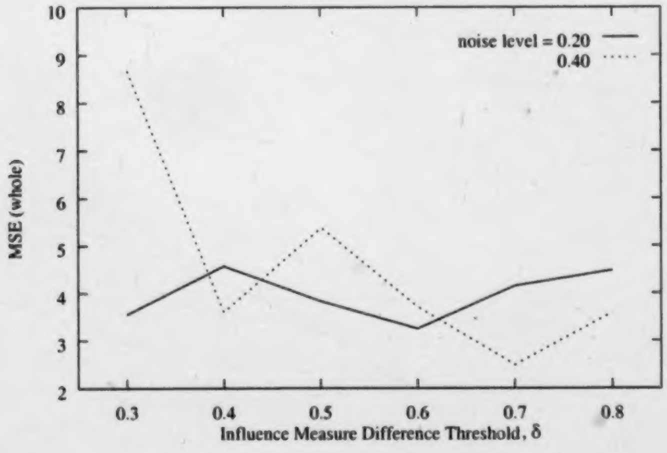


(b)

Figure 2.20: The semi-rigid models perform consistently better than the original balloons. Even when both of them cannot reach a stable state (noise level ≥ 0.8), the semi-rigid models produce results with a much smaller MSE. The gap size = 30.



(a)



(b)

Figure 2.21: Effects of parameter δ. The gap size is 30, and the normal force = 0.35. When $\delta > 0.4$, the algorithm performs well consistently.

Chapter 3

New Internal Energy Models

3.1 Problems with the Balloon Models

The original Active Contour Models are somewhat myopic, *i.e.*, they are not able to find distant boundaries and need a close initialization to start with. Cohen and Cohen's balloon models [14] addressed this problem by employing a pressure force to make themselves expand, thus searching a larger area. The balloon models are the first that adopt the idea of *active search*, as opposed to being passively attracted to boundaries. However, there are a few problems that the balloon models have not addressed.

The first problem is related to the contracting forces intrinsic to the original models. As illustrated by Gunn and Nixon [23], the contracting forces produced by the internal energy are not scale invariant because they decrease as a snake shrinks. Moreover, they are affected by the time step [23] and other factors, such as the image potential. In summary, the contracting forces are difficult to predict and vary both along the contour and along the time line. This fact makes it very difficult to choose an appropriate inflating force. To solve this problem, Xu *et al.* [57] proposed adding a new pressure force to totally offset the contracting forces, and their models are less sensitive to parameters. However, removing the contracting forces altogether also removes their ability to stay smooth while moving.

The second problem is with the re-sampling or re-parameterizing process of the ACMs. Unlike the original snake models, an initialization for a balloon model could be rather rough and far away from the object boundary. While expanding, a balloon needs additional snaxels to fill in gaps between sparse snaxel pairs. Common practice either re-parameterizes the curve and re-samples node points [33, 14], or adds snaxels to the most distant snaxel pairs [28]. However, both methods would disrupt the energy balance because most internal energy models use distance between neighboring snaxel pairs to assess their tension. A balloon relies on its tension to hold back snaxels with no image force to counter the inflating force. Therefore, maintaining tension while allowing additional snaxels to be added to a balloon is an important issue.

In this chapter, we present a multi-scale non-shrinking internal energy model as part of our Actively Searching Contour Models (ASCMs). The ASCMs are improved balloons and are designed with three objectives in mind: (1) Effectively avoid local minima while searching for the global minimum. (2) Perform contour completion for small gaps. (3) Provide an easy and reliable way to choose parameters. Our new internal energy model is another step forward to achieving these objectives.

3.2 An Unbiased Non-Shrinking Model

The original internal energy model consists of two parts: an elastic term that makes a snake behave like a string, and a stiffness term that makes it act like a rod. However, this technique results in the contraction of snakes. Removing the contracting forces entirely, as in [57], also results in reduced smoothing power. We argue that a smoothness constraint should not be over-powerful, but just powerful enough to serve its purpose, thus reducing its side-effects. Another preferred quality in a smoothness constraint is to have no prejudice towards expanding or contracting.

To make it more manageable and controllable, we divide the whole internal energy model into two parts: a smoothness part and a tension part. In this section, a new smoothness constraint, whose purpose is to keep a snake smooth and its snaxels equally distant, is presented. In the next section, the issue of how to maintain tension while introducing new snaxels is discussed.

A contour is considered smooth if every internal angle, ϕ, as shown in Fig. 3.1 (a), is within a smoothness range $(\pi - \theta, \pi + \theta)$, where θ is a small angle. In the case of a circle,

$$\theta = \pi - \frac{(N-2)\pi}{N} = \frac{2\pi}{N}, \tag{3.1}$$

where N is the number of snaxels of a discrete snake. Given the positions of snaxels x and y and the angle θ, two arcs can be constructed such that any point z inside the region enclosed by the two arcs will have an internal angle within the specified smooth range. This region is what we call a *smooth* region (Fig. 3.1 (b)).

Once a snaxel is inside the smooth region spanned by its two neighbors, it is considered within the smoothness limit. Otherwise, the snaxel is pulled towards the smooth region via the shortest route, *i.e.*, from the snaxel to its corresponding arc center. As shown in Fig. 3.1 (b), z is pulled towards c_1, one of the arc centers, provided that $\angle xzy$ is larger than π. Arrows indicate the direction and magnitude of the smoothing forces generated.

To compute the force vector **v**, the center of one of the arcs should be computed first. To make the formula simpler, we use location symbols to represent vectors, eg., x represents the vector from the origin, o, to location x. Assuming that $\angle xzy$ is less than π, c_1 and **v** are calculated using:

$$r = \frac{|x-y|}{2\sin(\theta)} \tag{3.2}$$

$$c_1 = \tfrac{1}{2}(x+y) + r\sin(\theta)\mathbf{RU}(x-y) \tag{3.3}$$

$$\mathbf{v} = (|c_1 - z| - r)\mathbf{U}(c_1 - z) \tag{3.4}$$

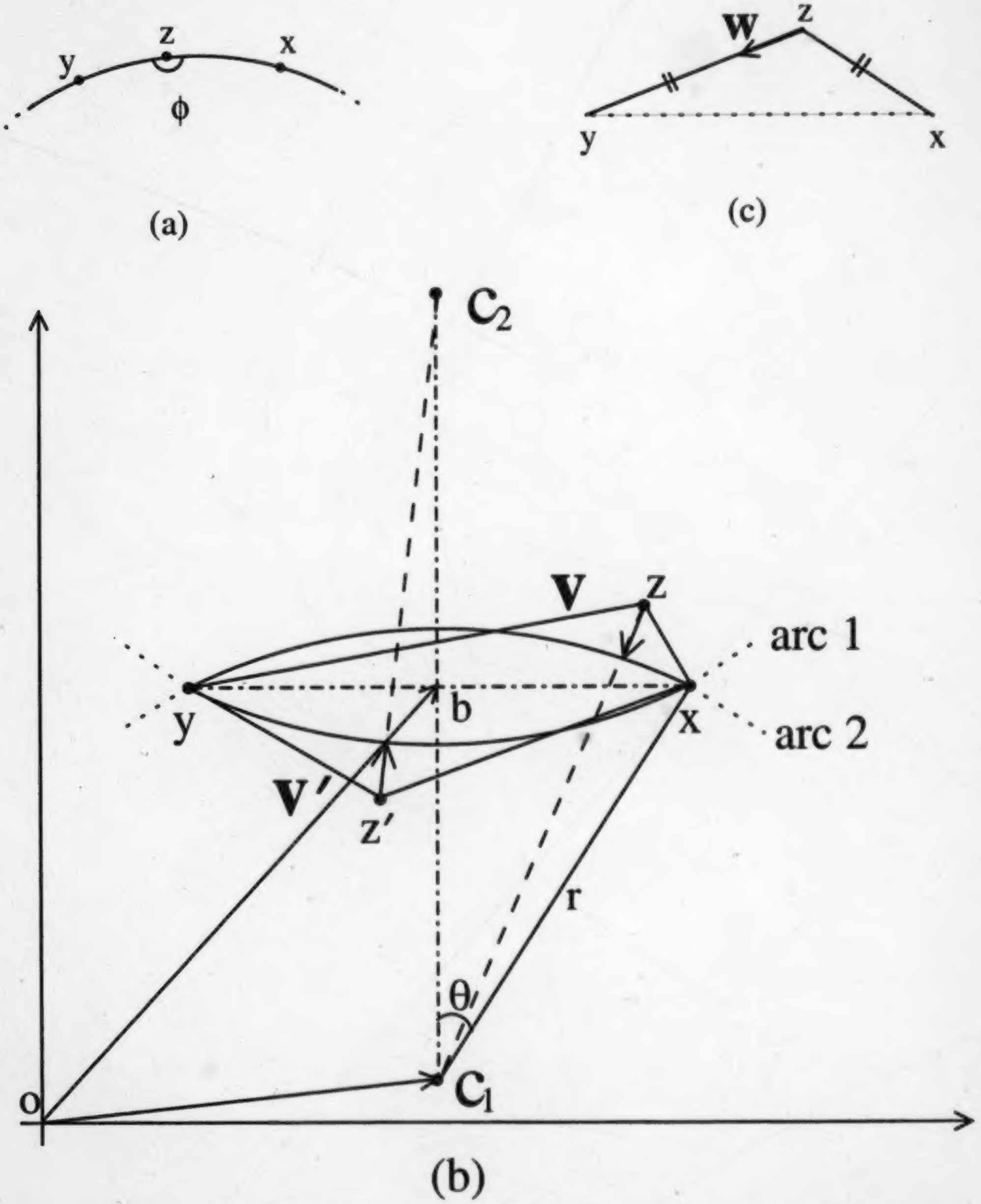
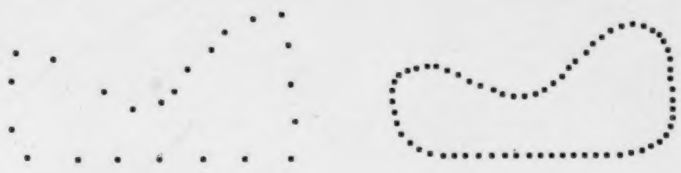

y
z
x
ϕ
(a)
w
z
y
x
(c)
C2
V
z
arc 1
y
b
x
arc 2
V′
z′
r
θ
o
C1
(b)

Figure 3.1: A new smoothness constraint: (a) the internal angle ϕ; (b) the smooth area (enclosed by the two arcs) and the two force vectors computed to pull snaxel z or z' to the smooth area; (c) another force vector $\mathbf{w}$ generated to maintain the equal distance property.



(a) (b)

Figure 3.2: An example using the new smoothness constraint: (a) an initialization; (b) the final shape after energy minimization with only the internal energies.

where **U** is the unit operator, and **R** is a 90° clockwise rotation operator. It should be noted that r is a scalar.

To ensure that snaxels are evenly spaced, a second condition is applied. z is pulled by another force **w** to whichever one of its neighbors is farther away (Fig. 3.1 (c)). **w** is computed using

$$\mathbf{w} = (|y - z| - |x - z|)\mathbf{U}(y - z), \tag{3.5}$$

and the total internal energy is normalized by the average distance, h, between snaxels.

$$E_{internal} = \frac{1}{2}\left(\frac{|\mathbf{v}|^2 + |\mathbf{w}|^2}{h^2}\right) \tag{3.6}$$

The smoothness constraint we just described has several important properties:

1. There are no contracting forces. In the absence of external forces, the proposed smoothness constraint will smooth a snake like a low-pass filter, and then the snake will hold its shape. Fig. 3.2 (a) shows an initialization and (b) the smoothed contour.

2. It is not biased towards expanding or contracting, and therefore it can be used along with a snake, with either an inflating or a deflating force.

3. In Fig. 3.2 (b), after being smoothed, the snake still has both concave and convex parts—a property that is missing in Gunn *et al.*'s model [23].

4. The internal energy is proven to be scale, rotation, and translation invariant (see proof in Appendix B).

5. The smoothness constraint can carry shape model information by specifying the internal angles for all snaxels.

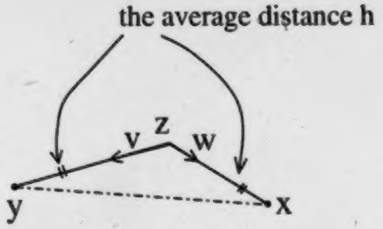


Figure 3.3: Two vectors **v** and **w** for tension computation. They are computed only when $|y - z|$ and $|x - z|$ are larger than the average distance h.

3.3 Maintaining Tension

A significant difference between a balloon and a traditional snake is the requirement for quality of initializations. A balloon is less demanding, and the initialization is generally only required to be inside the object contour. As described in Section 3.1, new snaxels are added to a balloon when it expands. This practice, however, is detrimental to maintaining a snake's tension, which is essential for a snake to perform contour completion for small gaps. Without enough tension, a balloon will go through even small gaps under the pressure of the inflating force.

Ivins *et al.* present a way to maintain the overall internal energy when inserting or deleting snaxels to/from a balloon [28]. However, the internal energy for individual snaxels may have changed even though the overall internal energy remains the same. In fact, the internal energy for individual snaxels is guaranteed to change because the total number of snaxels is increased, while the total energy remains the same. Those which relied on the tension from neighbors to maintain equilibrium before the addition of the snaxels, might start to move outward again. Such statements also hold for methods that resample node points.

We present a new procedure to minimize the influence of newly added snaxels on other snaxels. First, the method for computing tension is defined. We argue that tension should be applied only to necessary places. In line with this, we propose applying tension to snaxels whose distances to their neighbors are larger than the *median* or *average* value[1]. For snaxel z in Fig. 3.3 (a), its tension, T_z, is computed using:

$$\begin{aligned} \mathbf{v} &= (|y - z| - h)\mathbf{U}(y - z) && (3.7) \\ \mathbf{w} &= (|x - z| - h)\mathbf{U}(x - z) && (3.8) \\ T_z &= \frac{1}{2}\left(\frac{|\mathbf{v}+\mathbf{w}|}{h}\right)^2 && (3.9) \end{aligned}$$

where h is the preferred average or median distance between snaxels.

Now we assume that a snake with the new internal energy model is initialized to a perfect circle on a flat potential surface. Driven by an inflating force, the next state of the snake would be a concentric circle with a larger radius due to symmetry (Fig. 3.4

---

[1] For a deflating snake, those with closer neighbors are chosen instead.

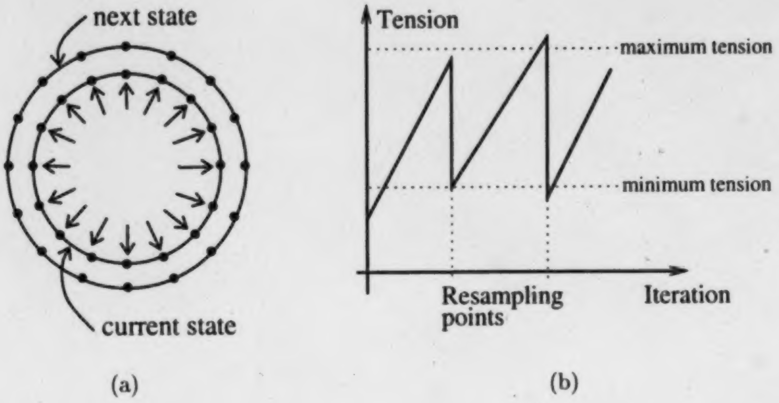


Figure 3.4: (a) under the influence of internal energies, the next state of a balloon initialized as a circle is a larger concentric circle; (b) a typical tension change pattern.

(a)). Since the snake maintains its smoothness, only the tension part of the internal energy has changed. The tension would increase linearly until new snaxels are added to the snake (see Fig. 3.4 (b)). Using the definition of a balloon, an inflating force should be large enough to overcome the maximum tension where there are no image forces.

The procedures reported in the literature choose positions to add new snaxels indiscriminately, or even simply resample the contour. This practice hurts the ability of a snake to perform contour completion at small gaps. We, instead, propose a heuristic search method to locate the most appropriate positions to add snaxels. Some criteria as to whether a location is appropriate are listed here:

1. Smooth—Adding a snaxel at a place with large curvature changes will prevent the smoothness constraint from taking effect, a weakness which is not preferred.
2. Average distance—If the distance between two snaxels is larger than the average, then the two snaxels are under the influence of a large tension force. On the other hand, if it is smaller than the average, then adding a new snaxel will further prevent them from developing tension.
3. Evenly distributed—It is preferable for newly added snaxels to be evenly distributed along the contour. In this way, energy change is evenly distributed as well.
4. Not on gaps—Adding a snaxel to a gap would reduce the tension of that part of the balloon and, possibly, the pressure force would drive the balloon through the gap. This is, therefore, not preferred. A procedure that identifies non-gaps has been proposed in Section 2.2.

One point to note is that adding new snaxels is done only periodically. A small period of time before the next try should be guaranteed, thus allowing a balloon

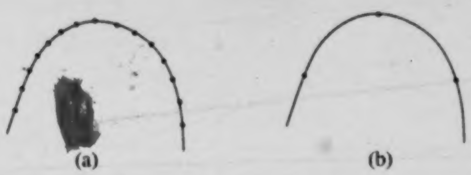


Figure 3.5: A snake discretized with different scales: it is considered (a) smooth using a small scale; (b) not smooth using a larger one.

to develop tension. To implement these criteria, a table is constructed whose entries contain, for each snaxel, the distance to its predecessor d_p, the distance to its successor d_s, curvature c, and a value g indicating if it is on a gap or not. All the entries are then sorted ascendantly according to the following **appropriateness** value:

$$a = \kappa_1(d_p + d_s) + \kappa_2 c + \kappa_3 g, \tag{3.10}$$

where κ_{1-3} are scaling factors. The first M entries are chosen as places where new snaxels may be added, except when one place is too close to some other newly added snaxels, in which case the entry is discarded. M is often chosen to be 5 – 10% of the total number of snaxels. The resulting reparameterization procedure is thus optimized according to the criteria listed above.

3.4 A Multi-Scale Internal Energy Model

3.4.1 Smoothness Constraint and Contour Completion

Contour completion is an important feature of the ACMs, and is essential to the extraction of contours of objects with fuzzy boundary. The heuristic search discussed in the previous section can prevent adding snaxels to gaps to some extent. However, a snake would still go through gaps if the tension is not built up. In this section, we examine the role of the smoothness constraint in contour completion.

Our theory is that the smoothness constraint is only effective for contour completion when the discretizing scale is comparable to the size of gaps. This is because, after discretization, the smoothness computed is directly related to the distance (scale) between consecutive snaxels. A contour may be considered smooth under one discretizing scheme but not under another, given the same internal angle for internal energy computation (see example in Fig. 3.5).

Some simulations are further provided to back up this statement. Figs. 3.6 (a), (c), and (e) show three partial balloons with the same discretizing scale, h, at three different sized gaps. The snaxels a, b, c, and d are assumed to be not moving, as if they were snagged by strong edges. All the partial balloons are applied with the same inflating force, and internal smoothing and tension forces. Figs. 3.6 (b), (d), and (f) show their equilibrium states under the influence of these forces. It is easy to see that larger gaps result in larger bumps. If new snaxels are introduced at these bumps,

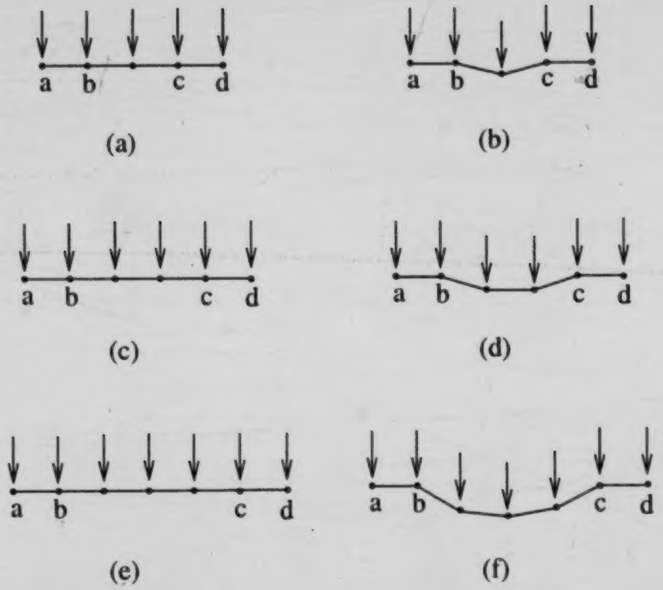


Figure 3.6: A simulation configuration: (a), (c), and (e) show three partial balloons with snaxels $a - d$ fixed; (b), (d), and (f) show their final shapes under the same inflating force. Arrows indicate the inflating forces applied to these partial balloons.

the balloons are ready to go through the gaps, thus losing their contour completion ability.

The average smoothness and tension energies are plotted in Fig. 3.7, which shows that the smoothness energy dominates the smaller gaps, while the tension gradually takes over as the gap size increases. This reflects the idea that discretization affects the smoothness energy computation, although the energy specification itself is scale-invariant after discretization.

3.4.2 Achieving Multi-Scale via Multiple Snakes

Since the discretization scale affects the ability of a balloon to perform contour completion, it is preferable to discretize a balloon with a scale comparable to the size of the gaps in the contours to be extracted. However, using too big a scale would also affect a balloon's ability to catch details. We present a multi-scale internal energy model to solve this problem.

A naive solution is to discretize a balloon at a scale that is small enough to catch any details, and then compute internal energies at different scales, *i.e.* using not only a snaxel's immediate two neighbors, but also those farther away. The fast algorithm presented by Williams and Shah [54] is able to perform the minimization using this scheme, with much more computation. For the more accurate minimizing algorithm, the time-delayed dynamic programming method presented by Amini *et al.* [2, 3], this increases the computation dramatically.

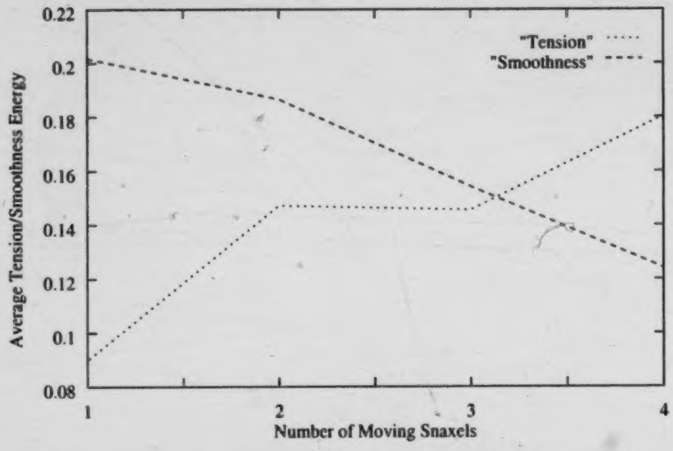


Figure 3.7: The average tension and smoothness energies for the three partial balloons shown in Fig. 3.6.

For example, if we consider four neighboring snaxels for every snaxel, the fast algorithm incurs twice the computation it normally needs. For the dynamic programming method, however, the computation is nm^5, in which n is the number of total snaxels, and m is the size of the neighborhood searched— compared to nm^3 if only two immediate neighbors are involved (see Appendix A for more details). If more snaxels are used, the algorithm's complexity increases exponentially.

In this research, we instead propose a multiple-snake method that takes advantage of dynamic programming's ability to use hard constraints. A balloon is still discretized at the finest scale that is required by application needs. A coarser snake is created by periodically copying snaxels from the originally discretized balloon[2]. More snakes are created, if needed, using the same method but with increasingly larger sampling periods. To make the snakes interact with one another, the corresponding snaxels are attached to one another using virtual strings (see Fig. 3.8). Each of these virtual strings has a natural length of zero and exerts a contracting force, when pulled apart, on the two snaxels it links:

$$E_{string} = \frac{1}{2}\eta \left|\frac{D}{h}\right|^2, \tag{3.11}$$

where D is the length of the string and η is a weighing parameter.

To make the balloon "feel" the image content, forces generated from the image are applied to each of the snakes. The inflating force is also applied to each and every snake, although the magnitude of the inflating force for different component snakes may be different.

The minimization is done sequentially for the multiple snakes. First, the energy for the first snake is computed, and all the snaxels are moved to their preferred next step. And then the same is done for the next snake, and so on. The process is

---

[2] The balloon with the original scale is called the *first* snake

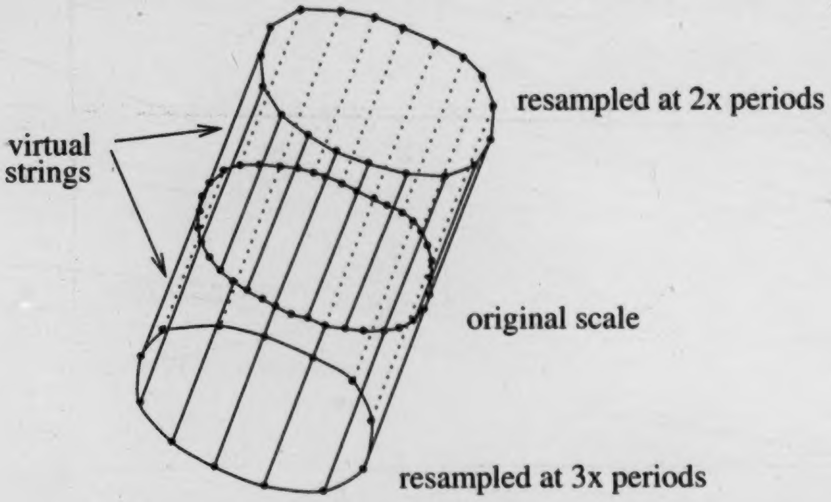


Figure 3.8: An example of a multi-scale snake. Three component snakes are shown in this configuration. The second and third snakes are attached to the first one with virtual strings.

repeated until none of the snakes is moving. The total computation depends on how many snakes are used, and how they are sampled. For the setup shown in Fig. 3.8, the amount of computation is $1 + \frac{1}{2} + \frac{1}{3}$ times that which the first snake needs. More snakes can be conveniently added with only a fraction of the computation.

3.5 Experiments

We designed several experiments to test the performance of the optimized reparameterization process and the multi-scale balloon models. The testing procedure introduced in Chapter 2 is again used. Since, this time, we are testing the contour completion function of the models, a new performance measure, MSE_{gap}, is introduced. To compute this measure, first the ground-truth circle and the balloon are aligned, and then the mean squared error of the snaxels on gaps is computed. Other performance measures introduced in Chapter 2 are also used. Again, in the following tests, balloons are stopped when their lengths exceed 130% percent of the ground-truth circle. The non-stable results are marked with a square in the subsequent plots, and should be viewed as solutions with *infinitely large* errors.

The second detail to note is that the rigidization process introduced in Chapter 2 is enabled in the following experiments, unless otherwise stated. In this way, the effect of the peeling problem is avoided. Since we are testing the contour completion ability of balloon models, it is essential that the expanding of balloons can be stopped by some mechanism.

The parameter η is adjusted in such a way that the resulting E_{string} is in the same order as other energies. A rule of thumb is that E_{string} should be adjusted

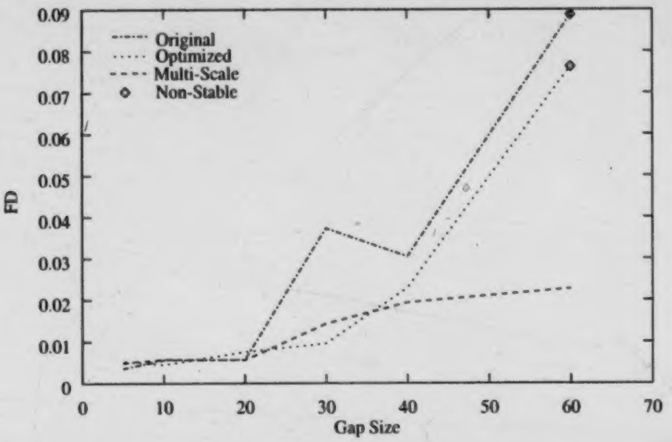


Figure 3.9: e_{FD} plotted against the gap size. The noise level is 0.0. Squares indicate non-stable solutions.

to be smaller than other energies, when the distance D is smaller than the average distance of the first snake. This is to ensure that the scale energy does not disturb the performance of other energies.

The first experiment tests the contour completion function of the optimized reparameterization process and the multi-scale models. Images with no noise, but changing gap sizes, are used in this test. The distance of the two sets of Fourier Descriptors, e_{FD}, is computed to measure the smoothness of the final result. Fig. 3.9 shows the computed e_{FD} plotted against the changing gap size. We observe that an optimized reparameterization method performs better than the original method. When the gap size grows too big (≥ 60), neither method can reach a stable solution. The multi-scale models that we used consist of two snakes; the first snake is discretized with the finest scale, and the second with double that scale. With the extra smoothness energy from the second scale, the multi-scale models perform consistently better. Even when the gap size reaches 60, the multi-scale models still produce very good results.

The second experiment is similar to the first one, except that the images are now added with zero-mean Gaussian noise with a standard deviation of 0.40. Four performance measures are plotted in Figs. 3.10 and 3.11. From these figures, we observe that: (1) The optimized reparameterization method generally performs better than the original method. Neither one can produce a stable solution when the gap size exceeds 50. (2) The multi-scale snake models perform consistently better than the single scale models, and they can produce stable solutions with larger gap sizes.

The third experiment uses a series of images with fixed gap sizes but changing noise levels, from 0.0 to 0.8. A single scale balloon with the optimized reparameterization process, and a multi-scale balloon, are applied to these images, and results are plotted in Figs. 3.12 and 3.13. The performance measure MSE$_{nongap}$ (Fig. 3.13 (a)) shows that the rigidization process performs equally well for both methods. All other performance measures indicate that the multi-scale balloon performs consistently better than the corresponding single-scale one.

An extra smoothness energy is useful, not only for contour completion, but also for avoiding local minima. Shown in Fig. 3.14 (a) is a common shape for a single-scale balloon. The upper concave part was caused by a small part of the balloon being trapped by a local minimum. The smoothness energy of the first snake is not enough to pull itself out of the local minimum. Usually, this is because the average distance between snaxels is set to a small number so as not to miss details. Fig. 3.14 (b) shows the first snake along with a second scale snake. For the second scale snake, escaping a local minimum is easier due to two reasons: (1) The same concave part is not "smooth" from the second scale snake's perspective. (2) A snake with loose snaxels misses some small details. According to the definition of a multi-scale snake, the two component snakes exert influence on each other (shown in Fig. 3.14 (b) as arrows). With this extra force, the first snake comes out of the local minimum and moves on.

Another experiment is done to show the noise-resisting property of the multi-scale models. First, a multi-scale balloon is applied to a test image with gap size 40 and noise level 0.4. During the iteration, the performance measure e_{FD} is computed and recorded. Then, a single scale balloon is applied to the same image with the same initialization. The values of e_{FD} are computed and recorded in the same way. The two sequences of e_{FD}'s are plotted and shown in Fig. 3.15, which shows that, after the first few rounds, the multi-scale balloon is smoother than the single scale one, reflecting the noise-resisting property of the multi-scale models.

A multi-scale balloon combines the advantage of two snakes with two different scales. The following experiment compares the performances of a multi-scale balloon and a single scale balloon. Unlike in previous tests, the scale of the single scale snake is equal to the scale of the second snake of the multi-scale balloon. The test image is again a circle but with numerous small gaps (shown in Fig. 3.16 (a)). Due to its larger average distance between snaxels, the single scale snake misses some details when expanding. A typical result produced by the single scale snake is shown in Fig. 3.16 (b). The multi-scale snake, however, produces very good results in a consistent way (Fig. 3.16 (c)).

The extra smoothness energy can sometimes be a disadvantage. Performing contour completion when necessary is an advantage, but a multi-scale balloon cannot tell when it is necessary to do so. It cannot discriminate between gaps and cusps or bumps. Fig. 3.17 (a) shows a contour with a gap, and a convex part with an opening approximately the same size as the gap. A result produced by the multi-scale models is shown in Fig. 3.17 (b), with both openings treated as gaps. A method that solves this problem is presented in the next chapter.

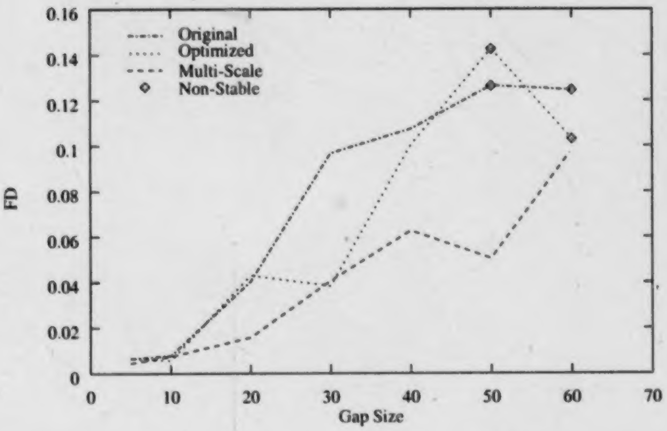
3.6 Conclusions

In this chapter, we first presented a new internal energy model that is scale, rotation, and translation invariant. Unlike traditional internal energy models, this new model does not shrink on its own, and is unbiased towards expanding or contracting. With the new internal energy, a snake would hold its shape—which may have both convex and concave parts—after being smoothed to some extent.

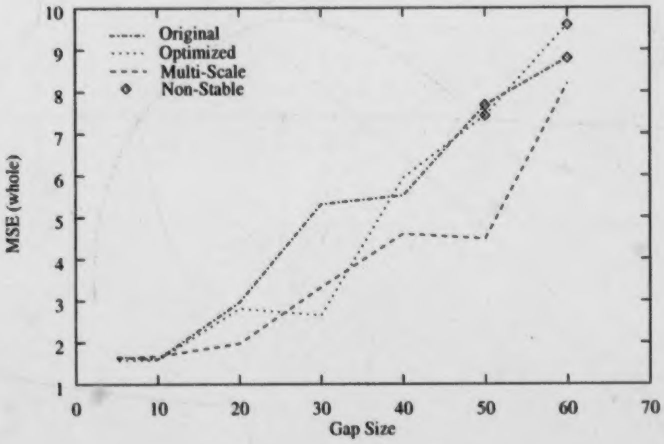
The ability of a balloon to perform contour completion is essential to locating incomplete boundaries. A better reparameterization process is thus presented to minimize the energy disturbance when adding new snaxels, although the performance improvement is limited.

To further improve the contour completion ability of balloons, a multi-scale balloon model is proposed. The idea of using multiple scales in computing internal energies is backed up by a simulation that demonstrates the relation between the smoothness constraint and the snake scale. To avoid excessive computation, we achieve the effect of multi-scale by using multiple component snakes linked to one another via virtual strings. Forces are generated from these strings to pull the component snakes together. The total computation is less than twice that for a single scale snake.

We implemented and tested the performance of the proposed methods using various test images. Results show that the new reparameterization method improves the performance of a balloon at gaps up to a certain size, but does not help a balloon with increased gap sizes in producing a stable solution. The multi-scale balloons, however, produce consistently better results with a larger range of gap sizes and noise levels. We also pointed out the new model's noise-resisting property and their inability to reach high-curvature areas. To conclude, we have provided a reliable and controlled way to perform contour completion.

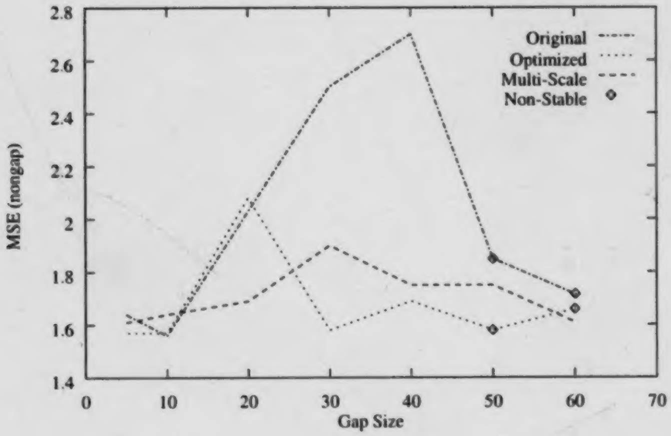


(a)

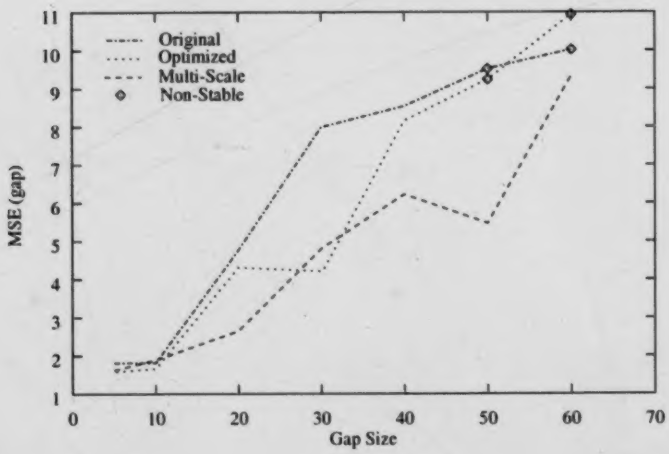


(b)

Figure 3.10: The test images are the same as those used in the first experiment, except they are now added with zero-mean Gaussian noise with a standard deviation 0.40. All four performance measures are plotted against the gap size. The multi-scale models can produce stable solutions all the time. Continued in Fig. 3.11.

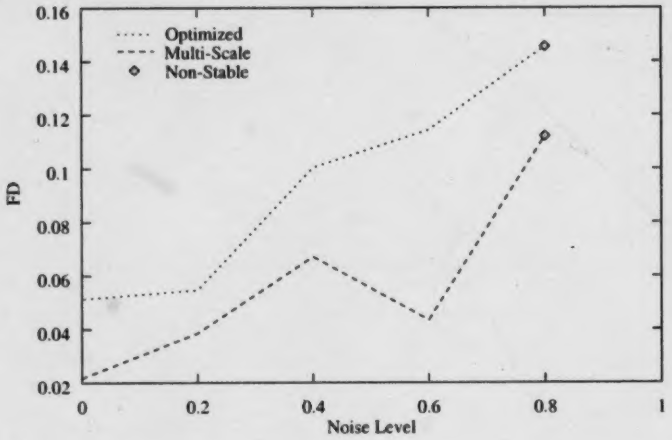


(a)

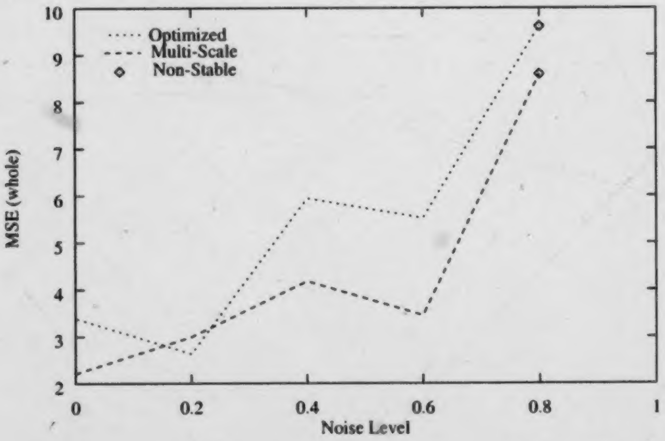


(b)

Figure 3.11: Continued from Fig. 3.10

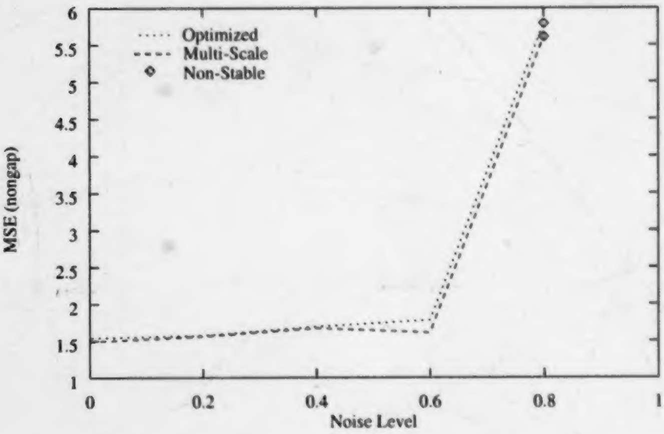


(a)

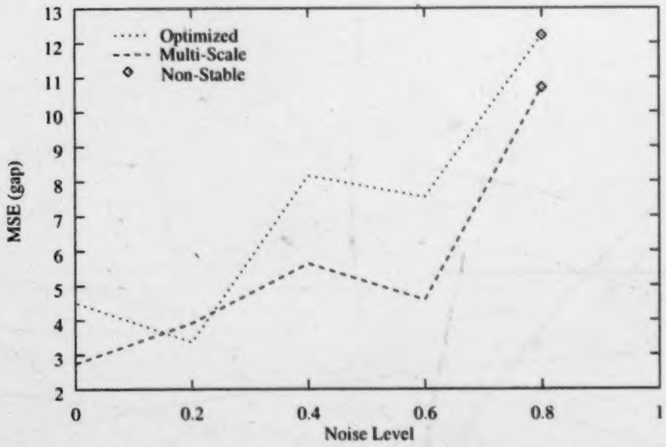


(b)

Figure 3.12: The gap size is fixed at 40, but the noise level changes from 0.0 to 0.8. All four performance measures are plotted against the noise level. Continued in Fig. 3.13.



(a)



(b)

Figure 3.13: Continued from Fig. 3.12. Figure (a) shows that the rigidization process performs equally well for both methods. The differences in other performance measures account for the performance difference of the two methods.

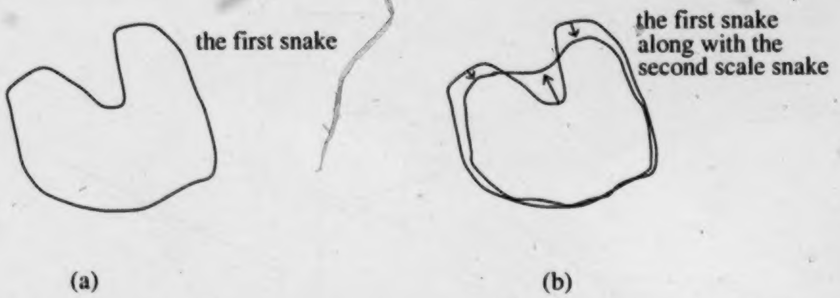

the first snake
the first snake along with the second scale snake
(a)
(b)

Figure 3.14: (a) a part of a balloon trapped by a local minimum; (b) forces shown as arrows are generated from the two component snakes.

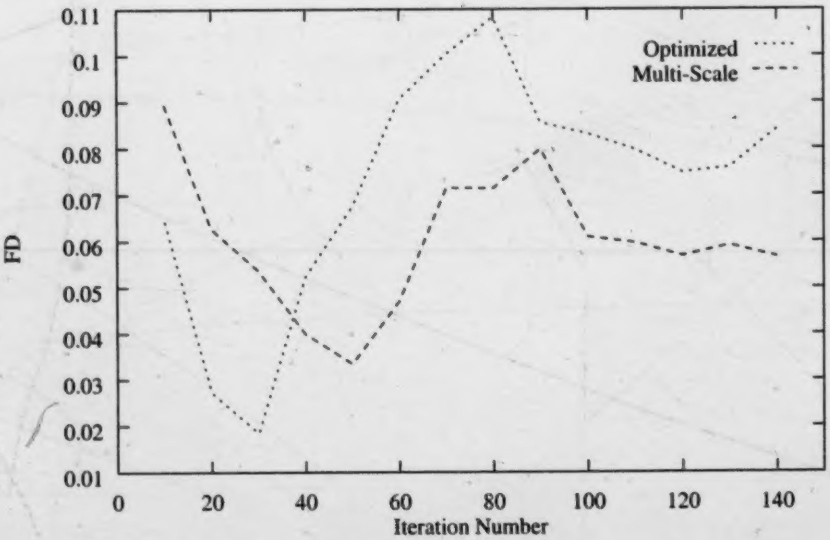
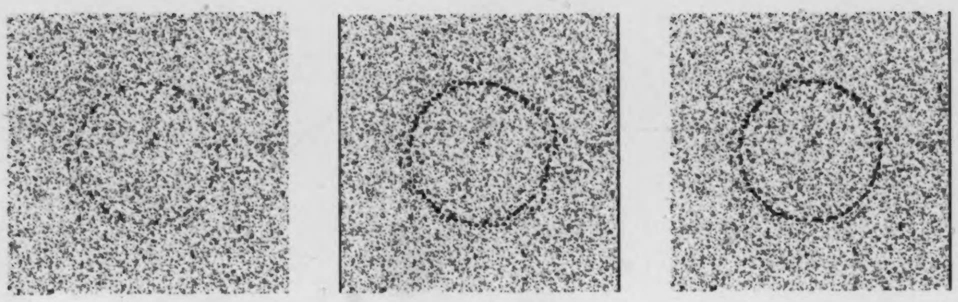
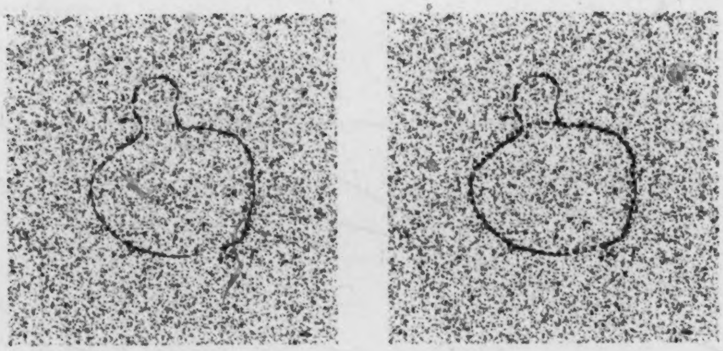


Figure 3.15: e_{FD} computed during iteration of a multi-scale snake with two component snakes, and a single scale snake. A smaller e_{FD} indicates that the balloon is smoother during the iteration.



(a) (b) (c)

Figure 3.16: (a) a circle with numerous small gaps and with noise level 0.4. A typical result produced (b) by a single scale snake; (c) by a multi-scale snake with two component snakes. Note that the single scale snake has the same average distance as the second snake of the multi-scale snake.



(a) (b)

Figure 3.17: (a) a contour with a gap and a convex part. (b) the result produced by a multi-scale snake. Both the gap and the opening to the convex part are treated as gaps, which is not true for the convex part.

Chapter 4

Content Guided Search

This chapter proposes another improvement to the balloon models' searching ability to reach high-curvature areas of the contour, which cannot be easily achieved by the semi-rigid models (Chapter 2) and the multi-scale models (Chapter 3). We propose here to use local image contents to guide a balloon's searching process. Sha'ashua and Ullman's saliency map method is used to trace the strongest curve that passes through a particular pixel. This locally derived information is then used to guide a balloon's searching process. Benefiting from both a global model and locally derived cues, the combined model is faster and more robust than the original balloon model.

4.1 Introduction

Active Contour Models [31] were presented as a "power-assist" to human operators for object contour extraction. In a typical scenario, a human operator draws a contour as an initialization for a snake, then the snake is progressively refined to match the actual contour. If we want to use snakes to automatically extract contours from images, however, the original snake models are not directly applicable due to the initialization problem. Cohen and Cohen's balloon model [14] uses an inflating force to make a snake expand until it meets an object boundary. The advantage of using a balloon model resides in its ability to search for contours from a rough, far away initialization. However, the balloon model is primitive in terms of its searching power. Moreover, since it starts with a rough initialization, the starting shape does not contain much *a priori* shape information.

As discussed in previous chapters, three problems need to be addressed to make balloon-based models effective: (1) How to perform contour completion when necessary? (2) How to overcome local minima? and (3) How to provide an easy and reliable way to choose parameters? The first problem is extensively addressed in the previous two chapters. By providing a new multi-scale internal energy model and a semi-rigid model, we have provided an effective way to make balloons perform contour completion. However, these two methods do not have a way to differentiate gaps from bumps and cusps, thus preventing a balloon from converging to those high-curvature areas. We argue that information derived from the underlying image should be exploited to make the balloon discriminative. The image content information we try to incorporate is what we call "mid-level" image cues, *i.e.* they are outputs of some mid-level processing modules instead of raw image data such as intensity and edge. In next section we provide more background information on the reasons why mid-level image cues are important, especially for noisy images.

4.1.1 Method Overview

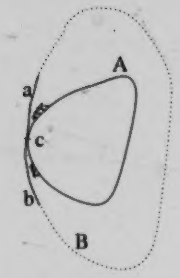
As discussed in Chapter 1, boundary extraction approaches in literature can be divided into two categories. The first category methods, which are called bottom-up methods, employ a two-step framework: Edge detection followed by edge linking. Since the result of an edge detecting operation is usually a set of spurious edges with many gaps, it is necessary to have an edge aggregation process after an edge detection operation. A large number of papers have been published in this area [37, 58, 18, 1, 44]. These methods share the property of using neighboring pixel information to help label an image pixel. They exploit the fact that noise points do not have support from their neighbors to suppress noise and reinforce detected structures. However, these methods can only deal with small gaps since there is no global shape model involved in the process.

On the other hand, the top-down methods feature exploitation of knowledge. Rule-based methods, scene labeling, deformable models, and active contour models [25, 31] are in this category and have achieved considerable success. The top-down methods start from a high-level scene model and find evidence at pixel level to support the model. However, the top-down methods are usually computation demanding.

In this chapter, we combine the balloon model with a bottom-up method so that the combined model would inherit advantages of both methods, which are from two different categories. As discussed before, our ultimate objective is to use balloon models to automatically extract contours from noisy images. A balloon model is suitable for this task due to the ease of initialization. However, since the inflating force that drives the balloon model is artificial and isotropic, it is not able to sense changes in underlying image intensities. That is, it does not have an intelligent way to steer the balloon towards salient image features. We propose to incorporate an edge linking-like method, which can provide, for each pixel, some information about its neighboring area. If this information is effectively used by a balloon model, the resulting model is expected to be faster and more robust.

An illustration of our combined method is shown in Fig. 4.1. For each snaxel on a balloon A (*e.g.* *c*), the strongest curve that passes through it, is traced (shown as curve a-b). If the curve is strong enough according to some predefined threshold, it is used to guide the search of a balloon in this neighborhood. The arrows in Fig. 4.1 show forces generated to pull balloon A to desired contour B. In next section, we discuss how to use an edge aggregating method to trace curves, and in Section 4.3, how to guide balloon searching is presented.

The most important feature of the combined model is, unlike most snake models reported in literature, its effective use of mid-level image features. Chiou et al. used



(a)

Figure 4.1: An illustration of content guided contour search: Arrows indicate forces generated from local image content.

the neural network method to assign pixels to various categories for medical images, and the outputs are then fed into a stochastic active contour model for further processing [13]. Their method is designed specifically for one kind of images and it is not suitable for most other applications. Our method, however, is more general and is considered an improvement to ACMs themselves.

4.2 Tracing Curves

Sha'ashua and Ullman's saliency map (SM) method [44] was originally proposed to detect globally salient structures in an image. Simply put, the SM method extracts certain shapes such as smooth, long curves that attract our attention more than others. It is suitable for our task because of its three significant merits: (1) It is fast and the amount of computation is only proportional to image size regardless of image complexity; (2) The computed results contain information necessary for curve tracing; (3) Thanks to its iterative process, the computed results are more accurate and reliable than raw image data. This property is especially important when dealing with noisy images.

Technically, a saliency map is computed in the following way. For every pixel in an image, we consider a set of k "orientation elements" connecting the pixel to its neighboring pixels. An orientation element is either called an *active* element if it lies on an edge, or a *virtual* one or *gap* if not. At each point P, we check every connected sequence of orientation elements $p_i, \cdots, p_{i+N}$ starting from P, with each element being an active or a virtual one. The sequence of $N+1$ orientation elements, or a curve Γ of length N, is assigned a saliency value of

$$E(\Gamma) = \sum_{j=i}^{i+N} \sigma_j \rho_{ij} C_{ij}, \tag{4.1}$$

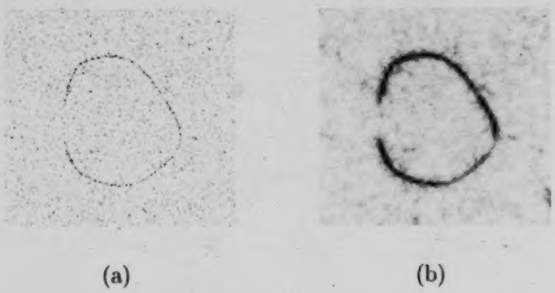

(a) (b)

Figure 4.2: A saliency map example: (a) a contour with a gap; (b) its computed saliency map.

where σ_j is a switch function indicating whether p_j is active or not and

$$\rho_{ij} = \prod_{k=i}^{j} \rho_k = \rho^{g_{ij}},$$

where ρ_i is a penalty factor. ρ_i is set to 1 if p_i is active, and otherwise is set to $\hat{\rho}$, a pre-defined constant typically ranging from 0.3 to 0.7. g_{ij} is the number of virtual elements between p_i and p_j. Finally,

$$C_{ij} = e^{-K_{ij}} \text{ with } K_{ij} = \int_{p_i}^{p_j} \kappa^2(s)ds,$$

where $\kappa(s)$ is the curvature at position s. K_{ij} is the accumulated squared curvature from the beginning of the curve and is used to penalize curving boundaries.

The saliency of an edge point P is defined as the maximum saliency of all curves emanating from it. A saliency network was proposed by Sha'ashua and Ullman to compute the saliency values. For each element p_i associated with a point P, we maintain a state variable E_i and a set of three attributes that include its local saliency σ_i, orientation θ_i, and penalty factor ρ_i. Each element p_i updates its state variable E_i iteratively through a local computation:

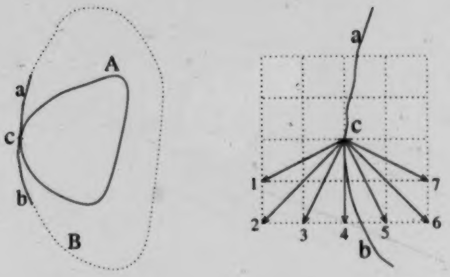
$$E_i^{(0)} = \sigma_i, \; E_i^{(n+1)} = \sigma_i + \rho_i \max_{p_j \in \delta(p_i)} C_{ij} E_j^{(n)} \tag{4.2}$$

where $\delta(p_i)$ is the set of k possible neighbors of p_i. It is proven in [44] that

$$E_i^{(N)} = \max_{\gamma_i} \sum_j \sigma_j \rho_{ij} C_{ij},$$

where γ_i is the set of all possible curves of length N starting from p_i.

An example of a saliency map is shown in Fig. 4.2. Fig. 4.2 (a) shows a hand-drawn image with Gaussian noise added and (b) its saliency map. Note that isolated noise points get small saliency values and small gaps are filled in.



(a) (b)

Figure 4.3: Tracing curves: (a) c-a is the most salient curve that originates from *c*; (b) the starting point of the second most salient curve is picked from a set of 7 points at the opposite side of the first starting point.

Once a saliency map is computed for an image, it is easy to trace the most salient curve that emanates from every pixel because orientations are recorded as part of the computation results. For a given pixel, the neighbor from which the largest saliency is computed is chosen as the starting point of the curve to be traced. However, only one curve that starts from every pixel can be traced in this way. For example, suppose that at some point a balloon A meets an object contour B at *c* (Fig. 4.3 (a)). Curve c-a is traced by following the largest saliency value. However, it is also necessary that curve c-b is traced unless *c* is an end point of contour B. We devised the following method to locate a second starting point. As shown in Fig. 4.3 (b), the second starting point should be at the opposite side of the first starting point (one of the 7 end points labelled as 1-7 on the discrete grid), otherwise the two traced curves would most likely collide with each other. Naturally, the second starting point is chosen as the the one with the largest saliency value among the 7 seven end points.

The saliency map method is extensively used in this thesis to provide the snake models with reliable mid-level information. It is also suitable for pre-processing ultrasound images [53]. The idea and computation method described above is quite brief and readers are referred to [44, 1] for detailed analysis of this method.

4.3 Content-Guided Search

The curves that are traced as discussed in the previous section are the strongest in their neighborhood. Since the SM method uses an iterative way to accumulate information, it is robust against random noise. Moreover, smoothness is included as a constraint in saliency map computation, therefore the curves traced are reasonably smooth provided the parameters are well adjusted. In summary, these curves are a strong local cue for a balloon. Properly harnessed, they can not only speed up a balloon's searching process but also make it more robust.

The traced curves can be used with any of the available energy minimization methods. In this thesis, the dynamic programming method [3] is the primary method for minimizing the energy of a balloon for better performance. Curve tracing and force generating is performed for each snaxel before each round of computation. To avoid unnecessary computation, only snaxels with a saliency value larger than a predefined threshold are used as starting points for curve tracing. However, the threshold is relaxed during tracing to allow some saliency value variation. A second condition regarding starting points selection is that one of a snaxel's two neighbors should have a saliency value less than the threshold. This condition is to ensure that the traced curves are indeed not repetitive. Traced curves are limited in length to 3 to 4 times of the average distance between neighboring snaxels.

Before the traced curves are passed onto later steps, some "bad" curves are eliminated based on the following conditions: (1) A curve must be smooth; (2) It should not be too far away from the balloon; (3) It should agree with the snake in terms of direction. The first condition is enforced by making sure an integral of curvature over curve length is less than a pre-defined threshold. For the second and third conditions, a series of internal angles, θ, between the traced curve and the balloon are computed as shown in Fig. 4.5 (a). The angles are required to be less than a threshold, which is chosen around 45 degrees in our experiment. The procedure for computing the angles is described in Fig. 4.4.

Converting local cues into influencing forces for a balloon is straightforward. As shown in Fig. 4.1, forces that would pull a balloon towards the traced curves are generated for this purpose. For each snaxel (for example, c in Fig. 4.5 (b)), we only consider 3 to 4 of its closest neighbors at one side (shown in Fig. 4.5 (b) as $n1$, $n2$, and $n3$). For each of them, a corresponding point (d in Fig. 4.5 (b) for $n3$) on the traced curve is determined so that the distance between c and $n3$ is equal to that between c and d. The direction of vector n3-d is a good direction candidate for the influencing force of snaxel $n3$.

The magnitude of the force is determined based on two factors: The saliency value of the curve and the magnitude of the normal force used by the balloon. The force is designed to be proportional to the saliency of the curve. In our experiments, the force is also chosen to be of the same order as the normal force so that the new influence force is able to offset the normal force when necessary.

Finally, if a particular snaxel is under the influence of several curves simultaneously, we choose to use the largest influencing force. The composite of these influencing forces is also an option. But usually these influencing forces have approximately the same direction, thus the composite force would be a lot larger than the normal force, which would disturb the balance of forces.

4.4 Experiments

As claimed before, our method can speed up the energy minimization process of a balloon model. Furthermore, the method is more robust in such cases as bad initializations, noise corrupted images, etc. In this section, several groups of experiments

```
procedure ComputeAngles()
begin
    r=current snaxel;
    p=r;
    n=r;
    c1=traced curve 1;
    c2=traced curve 2;
    repeat
        p=p's predecessor;
        n=n's successor;
        /* for p */
        find p's closest point t1 on c1;
        find p's closest point t2 on c2;
        $t = argmin_{t=t1,t2}(distance(t,p))$;
        compute angle between vectors r-p and r-t;
        /* for n */
        find n's closest point t1 on c1;
        find n's closest point t2 on c2;
        $t = argmin_{t=t1,t2}(distance(t,n))$;
        compute angle between vectors r-n and r-t;
    until one of the traced curves is traversed;
end
```

Figure 4.4: Pseudo codes for computing angles between a snake and a traced curve. The angles are then used to verify the curve's suitability for content guiding.

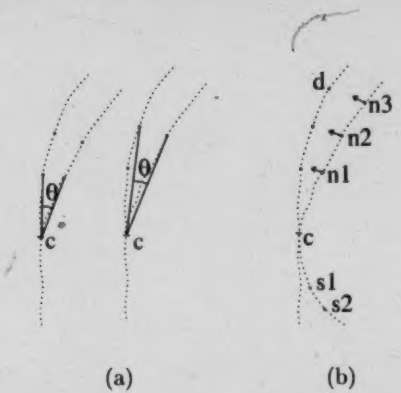


Figure 4.5: (a) a series of internal angles are computed as part of curve goodness measures; (b) several force vectors are generated to perform content guiding.

Figure No.	(a)	(b)	(c)	(d)
With the guiding module	84	80	70	120
Without it	112	98	103	150

Table 4.1: Iteration numbers for examples shown in Fig. 4.6.

are designed to test the performance of our method. In following experiments, the internal energy model proposed in Chapter 3 is used.

The first group of experiments test the speed of our method. Since the influencing forces are generated only when a balloon hits a strong curve, the speedup can only be observed when one part of a balloon hits a strong curve earlier than others. If all parts of a balloon meet a contour simultaneously, no speedup can be achieved. However, this is a rare scenario due to various reasons. A common reason is that a balloon is not initialized equally distant from the contour being sought. Even if it is perfectly initialized, some parts of a balloon can be snagged temporally by noise points. In this experiment, we test our method with some typical shapes with different initializations (Fig. 4.6). First, a balloon model is used to extract the contour in these figures, then the exact balloon model with the guiding model enabled is applied to these figures. Both programs are instrumented to record iteration numbers, which are listed in Table 4.1. These numbers, along with other experimental results we gathered, indicate a consistent speedup from our method.

The second test is on the robustness of the new method. Specifically, we test how this new method works with bad initializations and noisy images. For applications that require automatic extraction of contours, initializations are provided by a preprocessing program. If an initialization is not within the contour being looked for, a balloon model is not able to handle that. An example is shown in Fig. 4.7 (a). Since the balloon crosses the contour, some influence forces are generated from the two cross-points, A and B. Now if the influence forces are adjusted to be slightly larger than the inflating force, the outside part of the balloon would retreat and find the right contour (Fig. 4.7 (b)).

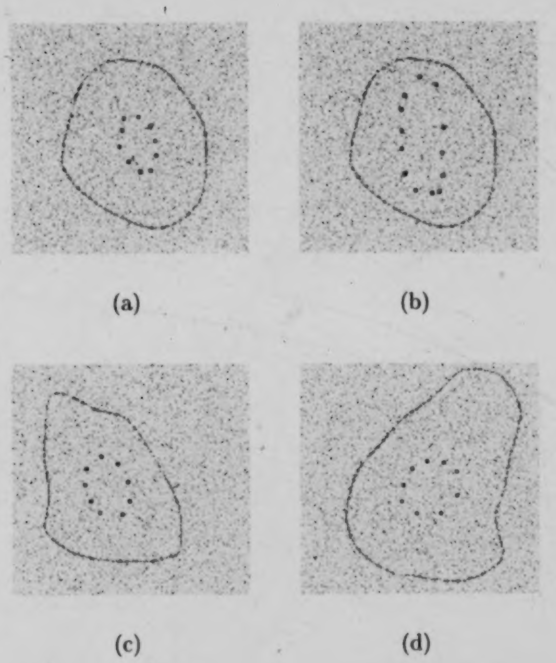
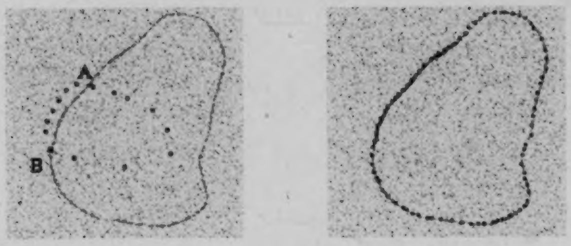


Figure 4.6: Some typical shapes with different initializations.

Another example of the robustness test is shown in Fig. 4.8. It is easy for a balloon to miss the boundary when approaching a circle with numerous gaps (refer to Fig. 3.16). For a contend-guided balloon, once one snaxel finds a piece of the circle, its neighboring snaxels are "informed" and know where to find the rest of the circle. Another problem with the original balloon model is, a snaxel may be pushed over the boundary by internal forces if it is on the tip of a cusp. Once it is pushed over, there is no chance for it to come back due to the inflating force. The content guided balloon solves all these problems and thus is more robust.

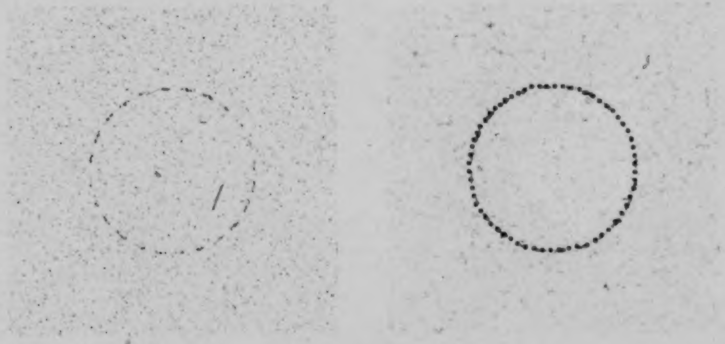
When a balloon is applied to an image corrupted by random noise, it is difficult to choose the weight for parameters, including the inflating force. Fig. 4.9 (a) shows an irregular shape immersed in noise. For the original balloon model, if the stiffness energy is given a large weight, then the balloon is not able to reach the upper-left corner (Fig. 4.9 (b)). On the other hand, if the stiff energy is reduced, then a typical result is shown in Fig. 4.9 (c). A perfect match could be achieved after careful adjustment of the parameters. For the content-guided searching snake models, however, this process is less painful. For both stiffness energy settings, our new method is able to extract the exact contour without any difficulties (Fig. 4.9 (d)).

To be more accurate on how tolerant the content-guided model is on noise, it is applied to a series of *unit* contrast circle images with various levels of noise. To get a feeling of how noisy these pictures are, an example is shown in Fig. 4.10 along with its computed saliency map. For comparison purpose, the original balloon model and the



(a) (b)

Figure 4.7: A bad initialization example: (a) an initialization; (b) the result produced by the content-guided snake model.

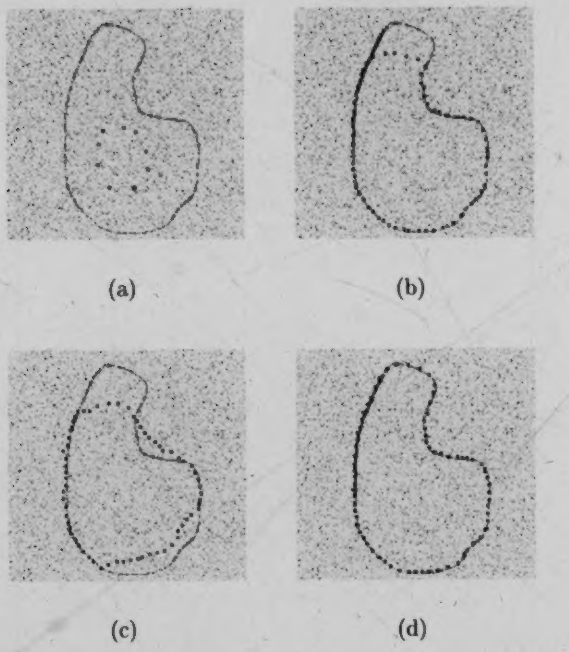


(a) (b)

Figure 4.8: (a) a contour with various small gaps; (b) the contour is successfully extracted by a content guided balloon.

semi-rigid model introduced in Chapter 2 are also applied to the same set of images. Again, the performance measures e_{FD} and MSE$_{whole}$, which are introduced in Chapter 2, are used. The results are plotted in Figs. 4.11. From the e_{FD} plotting, we can see that the original balloon cannot reach a stable state when the noise level exceeds 0.4. The semi-rigid model performs better, producing good results up to noise level 0.8. The performance of the content-guided model is even better and can tolerate noise level up to 1.2. This proves that the saliency map method is reliable even with high-level noise and the information it gathers helps make a balloon more reliable. At noise level 1.4, the e_{FD} value of the content-guided is larger than that of the semi-rigid model. This is because e_{FD} is a smoothness measure and with more parts attached to the circle, the content-guided model is not as smooth as the semi-rigid one.

The local information gathered by the saliency map method is enough for a balloon to tell a bump from a gap. The semi-rigid model and the multi-scale model introduced in the previous chapters have the ability to perform contour completion but cannot


(a) (b)

(c) (d)

Figure 4.9: Dealing with irregular shapes: (a) an irregular shape with an initialization; (b-c) two typical results produced by the original balloon models with different normal forces; (d) result produced by the content guided balloons with either normal force.

reach high-curvature areas. The content-guided model, however, can easily reach those areas with the help of local cues. Shown in Fig. 4.12 (a) is the image we used in Chapter 3 to show the semi-rigid model's problem. Fig. 4.12 (b) shows the result found by the content guided method.

4.5 Conclusions

A balloon is a global model because all snaxels have direct or indirect impact on one another. Moreover, a balloon can carry *a priori* shape information. It is a top down method that expands and converges to a contour that has the minimal energy on a potential field.

On the other hand, a bottom-up method such as the saliency map method gathers information from a small neighborhood. Through an iterative process, the information gathered is reliable and immune of random noise.

By combining the balloon model with the saliency map method, the content guided balloon inherits the power of both methods. Specifically, the balloon model combines image features, regularization factors, and knowledge-based constraints, while the saliency map method produces reliable local information. The combined model uses

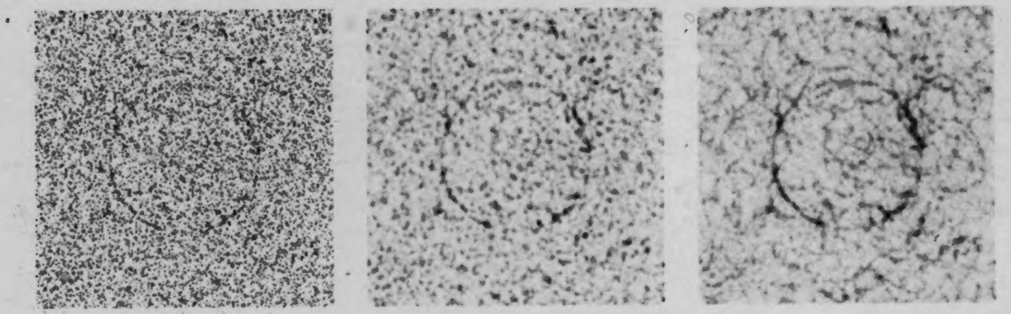
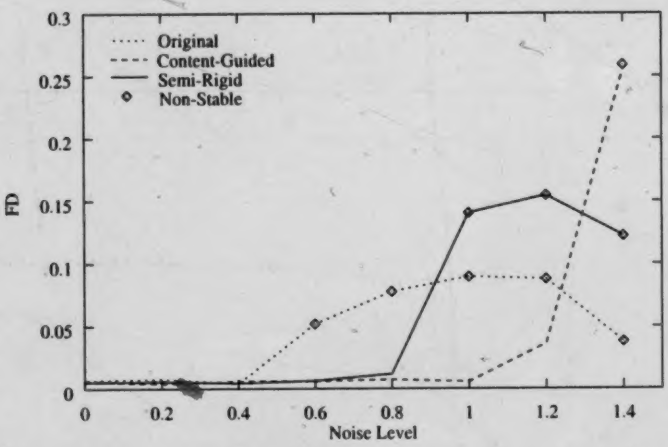

(a) (b) (c)

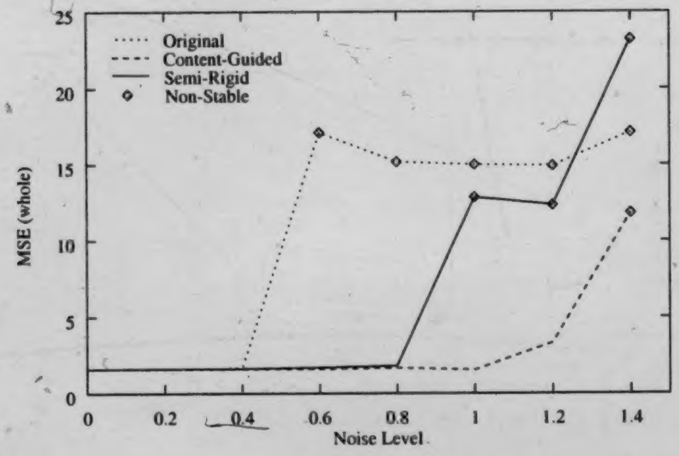
Figure 4.10: (a) an example test image with a noise level of 1.2; (b) the input edge image to the SM method; (c) the produced saliency map.

this information to guide a balloon's searching process. With this extra input, the improved balloon model is faster and more robust in many cases.

Experiments provided proves that the new method is robust against random noise, bad initializations, and other sources of misconduct by the original balloon. Furthermore, a content-guided balloon converges considerably faster due to its local steering ability. It can also reach high-curvature areas, solving the problem that plagues the semi-rigid and multi-scale models.

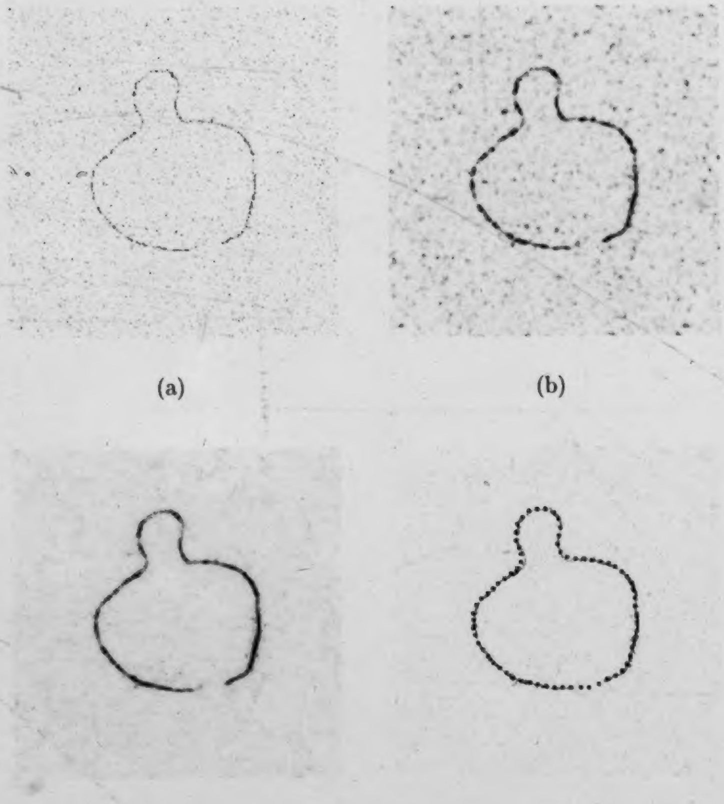


(a)



(b)

Figure 4.11: Testing the noise-resisting property of the content-guided model. The test image has a unit contrast circle with zero mean Gaussian noise. Noise level in this figure stands for the standard deviation of the Gaussian noise.



(a) (b)

(c) (d)

Figure 4.12: (a) a contour with both a bump and a gap; (b) the input image to the SM method; (c) the saliency map produced by the SM method; (d) the contour successfully extracted by the content-guided model.

Chapter 5

Controlled Accurate Search: A Combined Model

5.1 Introduction

In this chapter, we present several new ideas on how to accurately search for fuzzy contours. More importantly, these ideas are combined with those presented in the previous three chapters to form Actively Searching Contour Models (ASCMs). The resulting system can perform a controlled accurate search which, by our standards, should include being able to perform contour completion when needed, overcome the exact kind of edges specified, and provide ease of parameter choosing.

By accurate search, we mean that the strength of edges that a balloon is allowed to converge to, can be specified beforehand. During the iteration, a balloon would then overrun edges below the threshold and converge to those above the threshold. To achieve this objective, we propose two groups of methods.

The first group is aimed at providing the balloons with better potential fields. Since images are quite noisy, it is often difficult to differentiate legitimate edges from spurious ones, when using one single threshold. Even if a balloon has the ability to converge to edges with specified strength, it is not guaranteed that the edges the balloon converges to are all legitimate. To solve this problem, we propose a series of methods to pre-process noisy images so as to generate better potential fields, where the magnitudes of edges more accurately reflect their strength, and noise edges are greatly reduced.

The second set of methods consist of two parts. The first part improves the original balloon by using an adaptive inflating force. Unlike a constant inflating force, an adaptive one is computed locally for each snaxel, and is controlled by two parameters. The first parameter specifies the strength of edges a balloon should converge to, and the second one controls the inflation level of a balloon. An adaptive force is "just-enough" to overcome image forces, thus reducing the side-effects of using a constant one. In the second part of the method, a backtracking strategy is presented to facilitate the conducting of a multi-level feature search, which is complementary to the adaptive inflating force.

The rest of this chapter is organized as follows: Section 2 discusses how to prepare potential energy fields from noisy images; Section 3 introduces the adaptive balloon models; and Section 4 describes the backtracking mechanism. In Section 5, we present a global strategy to combine all the methods presented so far, to complete Part I of ASCMs.

5.2 Building up External Force Fields

In this section, we present a way to prepare external force fields, also called potential fields, for balloon models. The method was originally developed for ultrasound-like images, but is also suitable for any other images corrupted by random noise. It takes advantage of the fact that random noise does not have any support from its neighbors, and is based on the saliency map method introduced in the previous chapter.

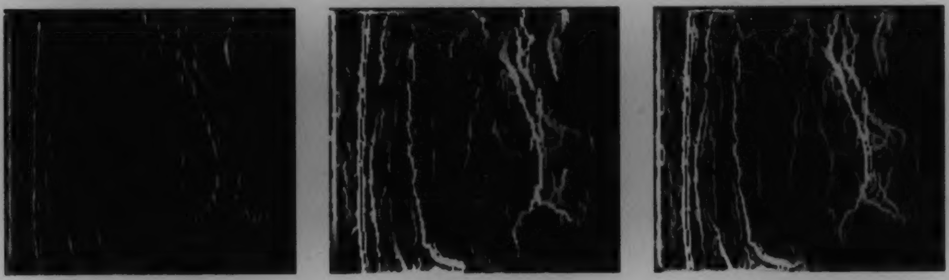
5.2.1 Rectifying Edge Magnitudes

Since one of our objectives is to apply the developed method to automatically segmenting ultrasound images, it is necessary to introduce some background on ultrasound images. Segmentation of ultrasound images has been plagued by so-called "ultrasound speckle", which is due to the physical mechanism of ultrasonic devices [7]. Ultrasound speckle noise is often modeled as a *multiplicative* process [7], *i.e.*, the speckle fluctuations in the signal are proportional in magnitude to the signal strength. This results in images with significant noise, even in very bright regions. Fig. 5.1 shows an ultrasound loin image, along with an edge image from a LoG operator. As shown in Bovik's analysis [7], the LoG operator can detect boundary edges accurately; however, it also produces a large number of erroneous edges due to speckle noise. Moreover, the erroneous edges in bright areas have a large magnitude—even larger than boundary edges in dark areas—for two reasons: (1) the multiplicative nature of speckle noise; and (2) the LoG operator (and most other edge detectors) searches for sharp intensity changes which are good only for detecting edges corrupted by additive noise. Boundary edges are immersed in erroneous ones, and this phenomenon poses a problem for most segmentation systems, since differentiating useful edges from erroneous ones is difficult.

It is necessary to rectify the edge magnitudes so that the side-effects from the multiplicative speckle noise are reduced. As observed by Czerwinski *et al.* [17], the boundaries in an ultrasound image have the appearance of straight or gently curving line segments because they are cross-sectional views of the surfaces between tissue layers. Based on this observation, we propose using the saliency map (SM) method, developed by Sha'ashua and Ullman [44], to transform the ultrasound edge images. As introduced in the previous chapter, the SM method uses a saliency network to compute a saliency value for every edge pixel, so that an edge pixel is assigned a large value if it is part of a long straight curve, and otherwise is assigned a small one. The transformed edge image is a better potential field for snakes, since the boundary edges are assigned a larger magnitude, and speckle edges are greatly reduced. Furthermore,

(a) (b)

Figure 5.1: (a) an ultrasound image; (b) an edge image produced by a LoG operator with $\sigma = 1.0$.



(a) (b) (c)

Figure 5.2: (a) the edge image produced by a Sobel operator; (b) the saliency map with $\rho = 0.5$ and $N = 20$; (c) the improved saliency map.

small gaps on boundary edges are filled in, leaving only larger gaps for the snake to cope with.

Since the saliency map method was designed to extract salient features from images, it works with intensity values directly. However, an edge image contains not only magnitudes, but also directions; therefore, it is necessary to revise the saliency map method to take advantage of the direction information as well. We propose encoding this information in the initial values of σ_i and ρ_i, for every orientation of a given point, p. To this end, we first apply an edge detector (*e.g.*, the Sobel operator) to the ultrasound image, and then normalize the edge vectors at each point. For each orientation element of a given point, the edge vector is projected to the particular direction, and the result is thresholded and assigned to σ_i. ρ_i is then assigned 1, if p_i is active; otherwise it is assigned the penalty constant ρ.

An example is shown in Fig. 5.2. The ultrasound loin image shown in Fig. 5.1 (a) is used as the test data. The edge image, which is computed using a Sobel operator, and the resulting saliency map are shown in Fig. 5.2 (a) and (b), respectively.

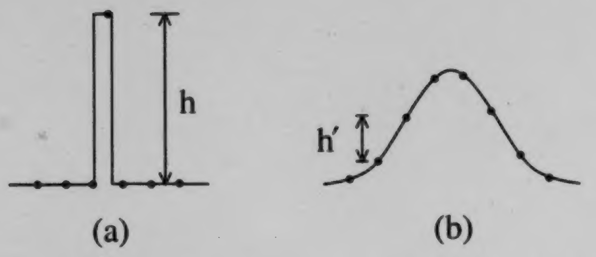


Figure 5.3: The potential energy related to the step edge in (a) is h, where that of the smoothed edge in (b) is h'.

It is easy to see that the saliency map is a better energy field for snakes, since the speckle noise is reduced and the boundaries are enhanced. We further propose improving the saliency map by calculating the saliency value for a given point p using

$$E(p) = \max_i(\rho_i E_i^{(N)} + \rho_{i'} E_{i'}^{(N)}) \tag{5.1}$$

where $p_{i'}$ is the element whose direction is the opposite of p_i's. In this way, the saliency value of a given point p is assigned the maximum saliency of all curves of length $2N$ on which p is the middle point. It should be noted that Equation 5.1 is only an approximation of the new definition. The result for the same ultrasound loin image is shown in Fig. 5.2 (c), and shows some improvements: boundaries are thinner and better connected.

5.2.2 Non-Maximum Suppression

The potential fields produced by the SM method also have some side effects. For example, a thin boundary becomes a thin band because neighboring non-edge pixels also get larger saliency values. The result is similar to an edge image after being smoothed by a Gaussian filter. Ideally, only the edge pixels with the maximum magnitude in the neighborhood should be marked as edges in the potential field. A saliency map is not really suitable for most of the minimization methods for snakes, mentioned in the literature. This is because, during the minimization process, only a small neighborhood (usually a 3x3 grid) is checked, and the difference in edge magnitude is the base for computation of the edge force. Due to the smoothing nature of the SM method, the edge magnitude difference between a pixel and its immediate neighbors is quite small, and does not reflect the strength of the original edge.

As shown in Fig. 5.3, a step edge produces an external force with strength proportional to its magnitude, h; however, the largest force produced by the smoothed edge is h', which is substantially smaller. The saliency map, then, actually produces a less effective potential field even though the magnitude of edges might have increased, and the noise has been reduced. Moreover, the position of the largest magnitude difference has also shifted, resulting in less accurate boundary location.

noise STD	0.40	0.60	0.80	1.00
before processing	0.162	0.242	0.325	0.403
after "AND" operation	0.156	0.193	0.254	0.263
after tracing	0.158	0.196	0.263	0.273

Table 5.1: Average noise intensity values before and after pre-processing.

We propose a two step post-processing method to reduce these side effects. The first step performs a thresholding procedure on the original edge image to obtain the position of edges. Then, an "AND" operation is performed between the original edge image and the saliency map. The magnitude of the pixels identified as edges by the thresholding process is replaced with the corresponding saliency value. Since a second step that traces edges from those already identified has been developed, the threshold-choosing process does not have to be very accurate.

The second step performs our version of hysteresis and non-maximum suppression. Using the saliency map, for each edge pixel, the two most significant[1] directions are derived, and are also traced for edge pixels not detected in the first step. This step is necessary because the SM method can perform gap completion for small or weak gaps, and these results should be included. To avoid introducing non-legitimate edge points, every new edge pixel traced should meet one more criterion: its most significant direction should point to the pixel it is traced from. In summary, these two steps perform thresholding with hysteresis, and achieve non-maximum suppression by not introducing non-maximum edges to the final results.

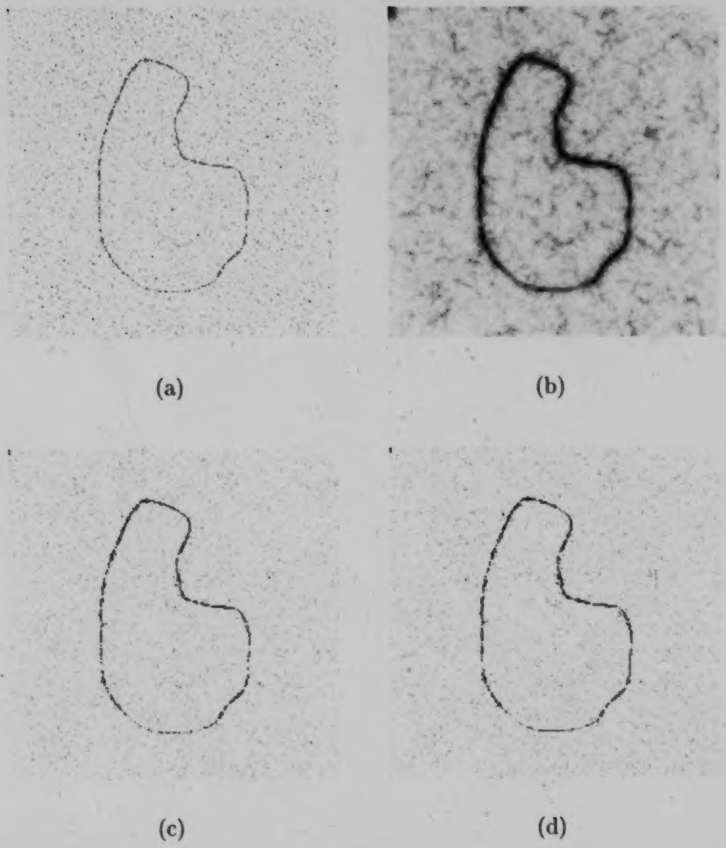
We use a synthetic image to demonstrate the power of our pre-processing method. Fig. 5.4 (a) shows a contour with a constant unit intensity. The synthetic image is then added, with zero mean Gaussian noise and standard deviations of $0.4, 0.6, 0.8$, and 1.0. Two of the processed results are shown in Figs. 5.4 (b)-(d) and Figs 5.5 (a)-(d).

Since the ground truth is a known contour with unit intensity, the average intensity values of the contour and background are used as performance measures. These values (computed before and after processing) are shown in Tables 5.1 and 5.2. We observe that the average values of the background noise decrease, while those of the contour increase—except for the last image, which is heavily immersed in noise. After the non-maximum suppression, the average contour intensity increases by 5.46%. Although the average noise intensity also increases, the level of 2.54% is much smaller.

Fig. 5.6 illustrates the processed results for the ultrasound image shown in the previous section (Fig. 5.1 (a)). Performance measures are not computed due to the lack of ground truth.

---

[1] The two most significant directions associated with each pixel: (1) the one with the largest saliency value; (2) the opposite of the first direction.


(a) (b)

(c) (d)

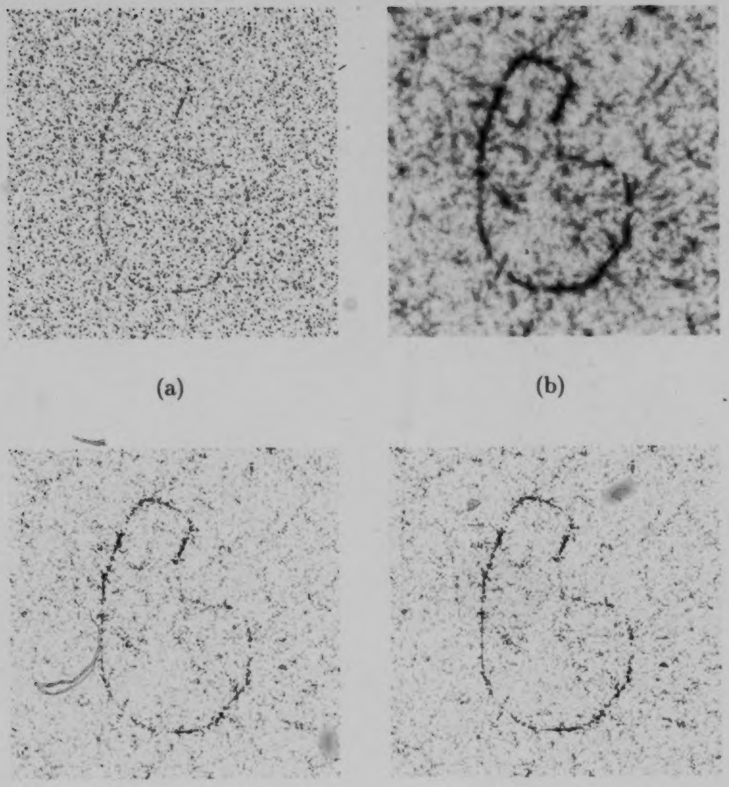
Figure 5.4: (a) a contour with unit intensity and a noise level of 0.4; (b) the saliency map; (c) after the AND operation; (d) after non-maximum suppression (tracing).

5.3 Adaptive Inflating Forces

For the original balloons, the inflating force and the internal smoothing force are expected to perform all the tasks: overcoming edges below a certain strength level, performing contour completion, and reaching high-curvature areas. The last two objectives are even contradictory to each other. The original balloons are not equipped with anything that can differentiate between gaps and bumps, and thus are not able to achieve both objectives at the same time.

Our solution for these problems is to dissect the general purpose forces into several specific ones, so that each of them performs one function. Below is a list of what has been achieved in the previous three chapters:

- Chapter 2 introduced a semi-rigid balloon model that provides extra stiffness force to counter the peeling effect. This reduces the instability of the energy minimization method, and it also helps in performing contour completion.



(a) (b)

(c) (d)

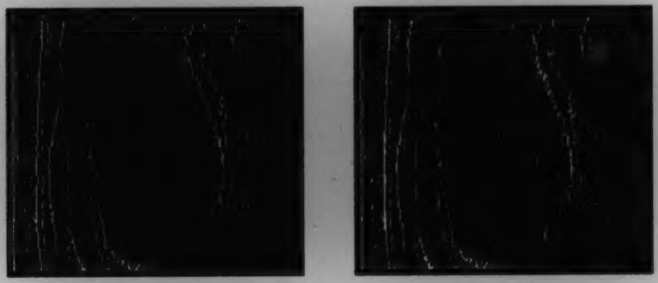
Figure 5.5: (a) another example with noise level 0.8; (b) the saliency map; (c) after the AND operation; (d) after tracing.

- Chapter 3 proposed an unbiased, non-shrinking, multi-scale internal energy model. This new internal energy model removes the contracting force without sacrificing its smoothing power. The multi-scale part of the model provides a controllable extra smoothness constraint, and is useful for contour completion.
- Chapter 4 presented a content-guided searching model which provides a way to reach high-curvature areas. Using information gathered from the image, the model is able to guide itself in converging to high-curvature areas where the smoothing forces are at their greatest.

In this section, we propose an adaptive inflating force to perform a controlled accurate search. First, problems with using a constant inflating force are discussed. In order to overcome the largest unwanted edges, and the contracting force from the internal energy, a constant inflating force is often set to a large value. Inevitably, a chosen constant inflating force is too large when encountering weak boundaries. At

noise STD	0.40	0.60	0.80	1.00
before processing	1.017	1.068	1.040	1.057
after "AND" operation	1.460	1.126	1.067	0.801
after tracing	1.501	1.163	1.131	0.855

Table 5.2: Average contour intensity values before and after pre-processing.



(a)

(b)

Figure 5.6: Results for the ultrasound image shown in Fig. 5.2: (a) after the AND operation; (b) after tracing.

gaps, this inflating force would push the balloon with its maximal strength, which partially causes the peeling problem. Since the contracting force is removed from our new internal energy model, an inflating force needs to provide a small force to overcome the tension and make the snake expand, and also to overcome image forces below a predefined threshold. Since the inflating force is no longer needed for reaching cusps and bumps, we propose using a "just-enough" adaptive inflating force, computed locally for each snaxel, to overcome the image force.

As discussed in Chapter 3, a balloon with the new internal energy model is easily pushed outward or inward: the tension does not have any effect if the distances between every neighboring snaxel pair are approximately the same as the average. With regard to smoothness, a parallel outward or inward shift of a snake part would not cause much energy change, as long as it stays smooth. In practice, the small force needed to make a balloon expand is easily decided by a few trials.

To perform an accurate search, it is necessary to be able to predict and control what levels of edges a balloon converges to. Now that an inflating force does not have to deal with a contracting force, the strength of the inflating force reflects the strength of edges a balloon will converge to. To overcome noise edges, an inflating force should be as large as the strength of noise edges in the image. A threshold computed from the statistics of an edge distribution is a good estimate of the strength of noise edges. Using this method, the inflating force can be computed systematically, instead of chosen empirically.

Traditionally, an inflating force is applied to every snaxel, regardless of what condition the snaxel is in. We argue that an inflating force should be used only to

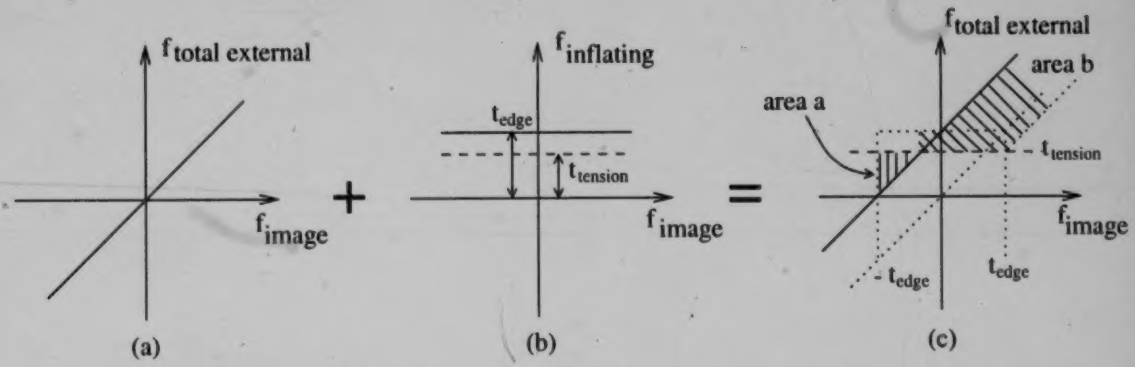


Figure 5.7: Using a constant inflating force: (a) the image force, f_{image}, is the only external force; (b) a constant inflating force as large as the noise threshold, t_{edge}; (c) the composite external force. Shaded areas indicate that the composite force is too large or too small.

assist a balloon in overcoming noise edges. When a snaxel is attracted by legitimate edges, it is not necessary to have an assisting inflating force. Even worse, it may be detrimental to the balloon, since the added inflating force can reduce the smoothing power of the internal energies. To illustrate this problem, Fig. 5.7 shows a constant inflating force added to varying image forces, where $t_{tension}$ is the minimal inflating force needed to overcome the tension, and t_{edge} is the edge threshold. For the shaded area a in Fig. 5.7 (c), the composite of the image force and the inflating force is less than $t_{tension}$, and, thus, is not enough to defeat the tension. For area b, however, the composite force is larger than $t_{tension}$ and would affect the internal energies' ability to maintain smoothness.

Removing the shaded areas from the composite force produces a more appropriate composite force. Fig. 5.8 (a) shows the desired composite force and (b) the adaptive inflating force, after removing the image force from the composite force. To summarize, the composite force is adjusted to be $t_{tension}$ when the absolute value of the image force is less than the edge threshold t_{edge}. For image forces larger than t_{edge}, which means the balloon is being pulled forward by the image forces, the inflating force is reduced to $t_{tension}$-t_{edge}. On the other hand, the inflating force is set to t_{edge} when the image forces are dragging the balloon, *i.e.*, when the image force is smaller than $-t_{edge}$.

By computing an adaptive inflating force locally for every snaxel, we ensure that each snaxel is only subjected to a minimal composite force, $t_{tension}$, when the image force is within a predefined threshold. By adjusting this force, we can control the inflation level of a balloon. On the other hand, adjusting the edge threshold, t_{edge}, controls the strength of edges that a balloon is allowed to converge to. These two parameters are separated from each other, and thus provide better control of the searching process.

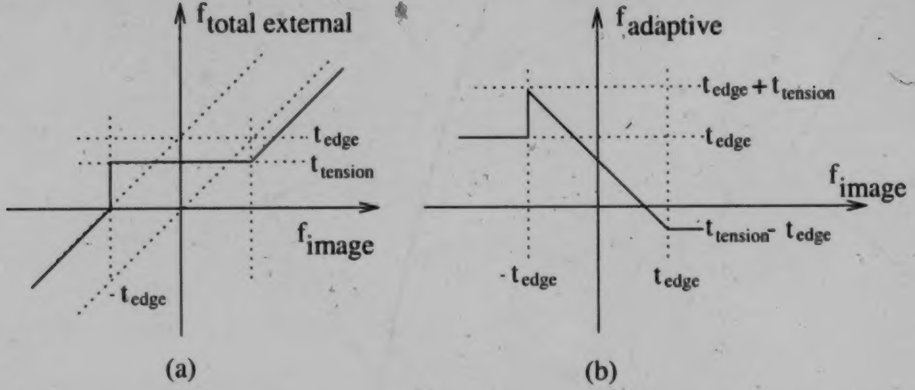


Figure 5.8: An adaptive inflating force: (a) the expected final external force after compensating for the insufficient part and removing the excessive part; (b) the adaptive inflating force plotted against the underlying image force.

5.3.1 Estimating Image Forces

The dynamic programming algorithm that minimizes the energy of a balloon uses a nine-pixel neighborhood, as shown in Fig. 5.9. Image forces are estimated from these nine pixels, and then the corresponding inflating forces are computed. To estimate the image forces, we first compute the normals to the contour (in both directions), which are shown as arrows and have components $(u1_x, u1_y)$ and $(u2_x, u2_y)$, respectively. For any $\mathbf{u}$, we consider the two points in the 8-pixel neighborhood of $P_{x,y}$ which lie closest to the line through $P_{x,y}$ in the direction of $\mathbf{u}$. For example, for $\mathbf{u1}$ in Fig. 5.9, pixels $P_{x,y+1}$ and $P_{x+1,y+1}$ are the closest ones to $\mathbf{u1}$. The gradient values of these two points are then used to estimate that of a new point in between them. The value of the interpolated gradient is

$$g_1 = \frac{u1_x}{u1_y} g(x+1, y+1) + \frac{u1_y - u1_x}{u1_y} g(x, y+1)$$

where $g(x, y)$ is the gradient value at the position (x, y). Similarly, the interpolated gradient at a point on the opposite side of $P_{x,y}$ is

$$g_2 = \frac{u2_x}{u2_y} g(x-1, y-1) + \frac{u2_y - u2_x}{u2_y} g(x, y-1)$$

The image forces are then calculated using the following formula:

$$\mathbf{f1}_{image} = (g_1 - g(x, y))\overline{\mathbf{u1}}$$

$$\mathbf{f2}_{image} = (g(x, y) - g_2)\overline{\mathbf{u2}}$$

where $\overline{\mathbf{u1}}$ and $\overline{\mathbf{u2}}$ are unit vectors.

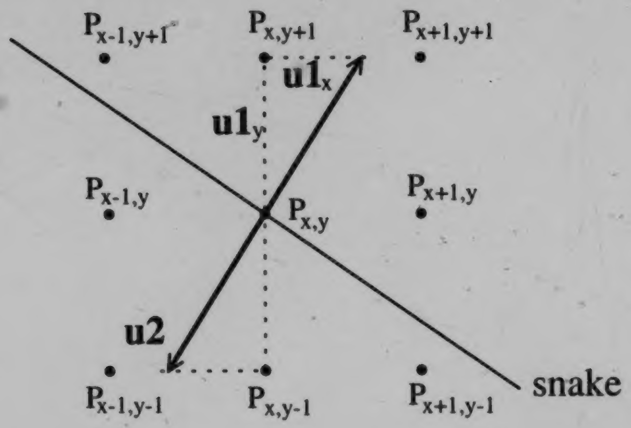


Figure 5.9: Estimating image forces for adaptive inflating force computation from a 3x3 neighborhood.

5.3.2 Backtracking

An adaptive inflating force gives the balloon models the ability to choose what levels of edges to converge to. Choosing an appropriate magnitude for the adaptive inflating force is also made easy. However, there is still a thresholding process involved in deciding how large the inflating force should be. For noisy images, it is inevitable that there will be some gaps or weak parts on the object boundaries. Although the adaptive inflating force is designed not to interfere with the function of the internal energies, the regularized solutions provided by the contour completion function are, at best, an informed guess.

Unless the gaps on the object boundaries are clean, there is generally some useful information that could be used. Usually, the edges are not totally missing only weaker than other parts. Thresholding with hysteresis is a common practice in edge detecting to take advantage of this fact. We instead propose a searching strategy, called backtracking, to look for weaker features.

The searching strategy is achieved by systematically changing the inflating force applied to a balloon. The force is chosen to be initially large and outgoing, but is flipped in direction, and reduced in magnitude, after reaching its first equilibrium. The process is repeated until the inflating force is small compared to other snake forces, *i.e.*, when a global minimum is found. The searching strategy makes it possible for a part of the snake to backtrack to areas previously visited and look for weaker features— a property that is crucial in the segmentation of noisy images.

An illustration is given in Fig. 5.10. The snake starts from contour A, and the object boundary of interest is B. There are some gaps along the boundary, as is the case for edges derived from real-life noisy images. A possible first-try result is shown as boundary C, which is over-inflated. The major factor that makes the snake stop at point C is the internal force from other parts of the boundary. Once it has reached the first equilibrium, the inflating force is changed in direction and reduced in magnitude;

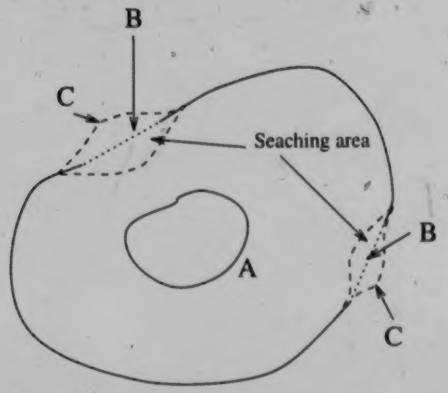


Figure 5.10: An illustration of the backtracking mechanism.

this makes the snake backtrack, and likely converge to weaker features.

5.4 A Combined Model

So far, we have presented a series of ideas that comprise our balloon-based ASCMs. These ideas were developed for one common purpose: to actively search for contours in a controlled manner, and with accuracy. In this section, we combine them into one framework that makes systematic use of these models. Fig. 5.11 is a diagram illustrating contour extraction using ASCMs. The process can be divided into two steps: the pre-processing step and the searching step. The pre-processing step includes edge detecting, saliency map computation, non-maximum suppression, and inflating force estimation. It is a preparation step that provides the searching module with a rectified edge map as a potential field, a saliency map for content-guiding, and an estimated starting value for the inflating force. Before the searching process is started, a rough initialization should be provided by either an operator or an initialization program.

The searching step is a loop that consists of three sub-steps: (1) energy minimization with all modules enabled; (2) continued energy minimization with only the content-guided module enabled; (3) parameter adjusting. This step performs the backtracking searching procedure described in the previous section. The first sub-step is to search for contours with all four modules enabled (see step [2.1] in Fig. 5.11). The inflating force is first set to the largest possible value estimated by the pre-processing step, and the multi-scale and semi-rigid modules are adjusted to have maximum effect. This is to ensure that the resulting balloon at the first equilibrium is over-inflated, setting the search area for later steps. The second sub-step is added to reduce the effects that the multi-scale and semi-rigid models might produce. For example, the smoothing power from the multi-scale model might prevent a balloon from converging to a sharp corner. By disabling these two modules, the balloon would dissipate the extra energy produced by them, and converge to sharp corners.

The third sub-step prepares for the next round of energy minimization. First, the

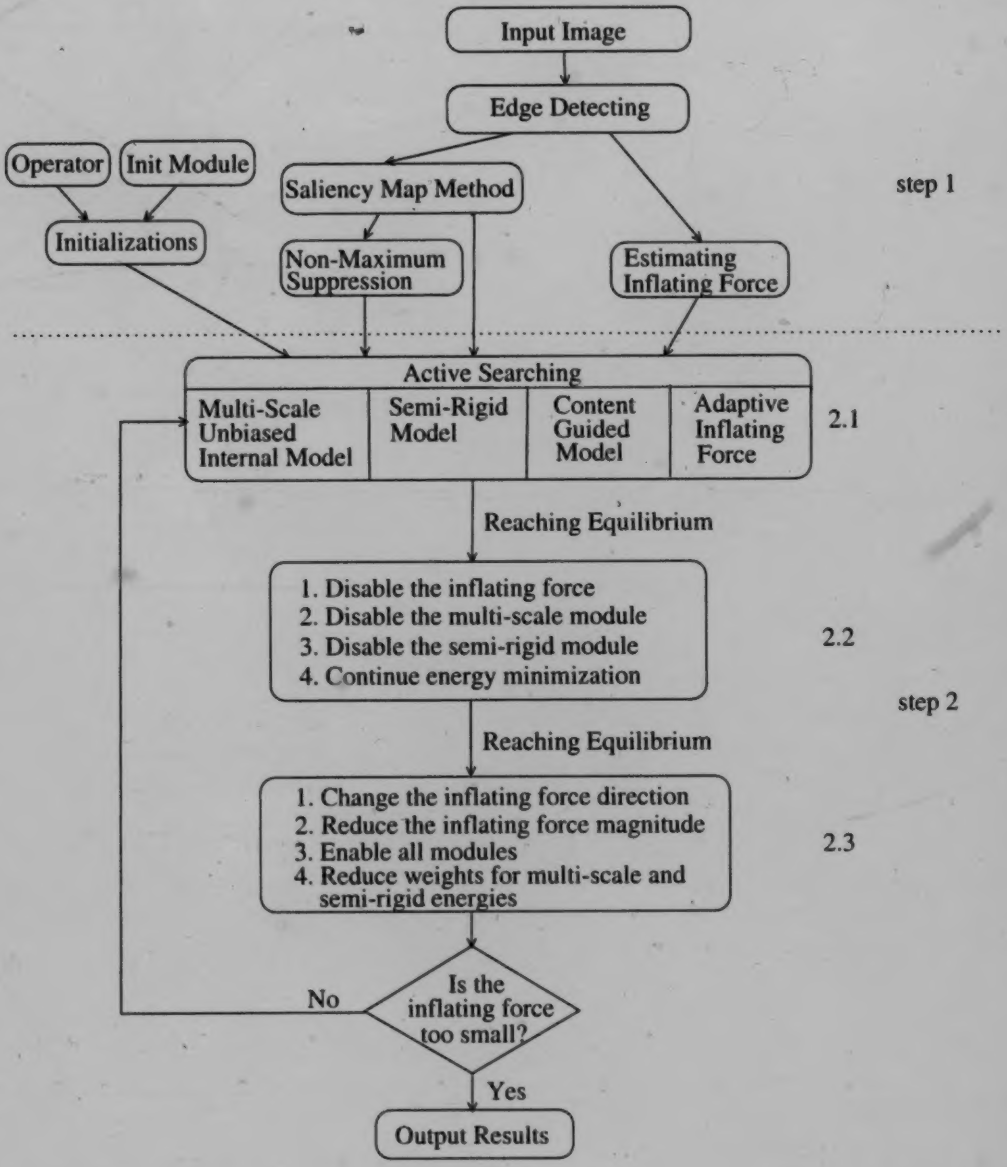

Input Image
Edge Detecting
Operator
Init Module
Saliency Map Method
step 1
Initializations
Non-Maximum Suppression
Estimating Inflating Force
Active Searching
Multi-Scale Unbiased Internal Model
Semi-Rigid Model
Content Guided Model
Adaptive Inflating Force
2.1
Reaching Equilibrium
1. Disable the inflating force
2. Disable the multi-scale module
3. Disable the semi-rigid module
4. Continue energy minimization
2.2
step 2
Reaching Equilibrium
1. Change the inflating force direction
2. Reduce the inflating force magnitude
3. Enable all modules
4. Reduce weights for multi-scale and semi-rigid energies
2.3
No
Is the inflating force too small?
Yes
Output Results

Figure 5.11: A diagram illustrating contour extraction using the Actively Searching Contour Models.

direction and magnitude of the inflating force are adjusted. The magnitude is reduced by a certain percentage, for example 10% to 20% every time. Since all modules interact with one another, others should be adjusted as well when the inflating force is changed. This is to ensure that the forces generated by these models have the same order in magnitude. One exception is the content-guided searching module. The weight for this energy is maintained at its original level because this force is generated from image content, and is reliable.

The parameters involved in these component modules can be adjusted in the third sub-step as well. For example, the threshold for the curve tracing method in the content-guided module can be lowered to allow tracing through weaker edges. Another important parameter to adjust is $t_{tension}$. Generally, this parameter should be reduced when the inflating force is reduced.

It should be noted that although there are a lot of parameters used in these component modules, choosing parameters is not difficult. We have demonstrated in the previous chapters that a fixed set of parameters works well for the same category of images. Also, for a specific parameter, a large range of values can be used, showing the robustness of the algorithms themselves.

5.5 Experiments

Two parameters, namely t_{edge} and $t_{tension}$, are used to control the adaptive searching process. The first experiment tests how $t_{tension}$ controls the inflating level of a balloon. As introduced in Chapter 2, synthetic images that we generate contain a unit-contrast circle with four gaps of controllable sizes (see examples in Fig. 2.15). In the first experiment, a circle with gap size 30 and zero noise level is used. The performance measures, MSE_{gap} and MSE_{nongap}, are again used to measure the closeness between a balloon and the ground-truth circle. MSE_{gap} measures the difference for the gap part of the circle while MSE_{nongap} measures that for the non-gap part.

In this experiment, an adaptive balloon is repeatedly applied to the testing image with t_{edge} fixed and $t_{tension}$ increased from 0.15 to 0.24, at an increment of 0.03. After one round of computation, t_{edge} is increased to a new level, and the process is repeated. Fig. 5.12 plots the performance measures, MSE_{gap} and MSE_{nongap}, against $t_{tension}$ for three t_{edge} levels, $0.20, 0.25$, and 0.30. From the MSE_{nongap} plot, we observe that, for the non-gap part of the circle, the resulting balloon is equally close to the ground-truth circle under a different t_{edge}. Switching to the MSE_{gap} plot, we can see that, for a specific t_{edge}, MSE_{gap} increases when $t_{tension}$ increases. This indicates that $t_{tension}$ controls the inflating level of the balloon. For different levels of t_{edge}, MSE_{gap} increases at a similar pace, and the values are in the same range, which means that t_{edge} does not affect the inflating level of the balloon. We conclude, therefore, that the parameter $t_{tension}$ alone controls the inflating level of a balloon.

The second experiment compares the performance of balloons using an adaptive inflating force with those using a constant force. The testing image in the first experiment is utilized again. For the balloon using an adaptive force, $t_{tension}$ is set to 0.15, which is the minimal force required. t_{edge} is increased from 0.15 to 0.30, with an increment of 0.05 each time. For the balloon using a constant force, the inflating force is set to the same value as t_{edge} used by the adaptive balloon. The performance measures are plotted in Fig. 5.13, where small squares indicate that the balloon cannot reach a stable state with parameters set to the given values. From this figure, we observe that a constant inflating force cannot produce satisfactory results if it is set too large. This is because the peeling problem occurs at the gaps and spreads to the non-gap parts. On the other hand, an adaptive force is equal to the minimal force $t_{tension}$ at gaps, which is substantially smaller than t_{edge}, and thus less likely to incur the peeling problem. This is further illustrated in Fig. 5.14, using a contour with a large gap on it. Fig. 5.14 (c) and (d) show two typical results of the traditional balloon, with the inflating force set to 0.22 and 0.30, respectively. Fig. 5.14 (b) shows the result produced by the adaptive balloon with t_{edge} set to 0.30. In conclusion, an adaptive balloon is less sensitive to parameter choosing, and is less likely to incur the peeling problem.

To test the accuracy of the adaptive searching process, a testing image with multiple concentric circles is generated (see Fig. 5.15 (a)). The intensity level of the circles increases by about 0.07 from inside out, starting from 0.24. An adaptive balloon is repeatedly applied to the image, with an initialization well inside the smallest circle. We set $t_{tension}$ to 0.15 and t_{edge} to various values from 0.16 to 0.64, with an increment of 0.02. Each time, the intensity of the circle that is extracted is recorded and compared to t_{edge}. Fig. 5.15 (b) plots the intensity of the circles extracted, against the given t_{edge} value. We can easily see that the actual results fit the expected ones quite well. We conclude that the parameter t_{edge} controls the strength of edges that a balloon is allowed to converge to.

To prove that the two parameters, t_{edge} and $t_{tension}$, are separated from each other, another experiment is conducted. In the testing image shown in Fig. 5.15 (a), four gaps are added to every circle. Each circle has the same percentage of its contour missing, with the third one having a radius of 50 and a gap size of 20 (see Fig. 5.16 (a)). For a traditional balloon, a typical result is shown in Fig. 5.16 (b), with the inflating force set to 0.35. Due to the peeling problem, the balloon converges to circles with far larger intensity levels. For an adaptive balloon with t_{edge} set to 0.35 and 0.41, respectively, the results are shown in Fig. 5.16 (c) and (d). From its performance at gaps, we conclude that t_{edge} does not control the inflating level of the balloon.

The following several experiments illustrate the benefits of the backtracking process. Fig. 5.17 shows that this process makes it possible for a balloon to find less strong edges. Fig. 5.17 (b) is the first equilibrium state of an adaptive balloon with the maximum t_{edge}. t_{edge} is then reduced and the direction is changed to the opposite one. After several rounds of backtracking, the final result is reached and shown in Fig. 5.17 (c).

Fig. 5.18 illustrates the importance of the second sub-step of the backtracking process. This figure shows a contour with sharp corners, which is hard to reach if the multi-scale and semi-rigid modules are enabled. In the second sub-step, the models are disabled to allow the balloon to converge to the corners shown in Fig. 5.18 (c).

The next experiment (Fig. 5.19) illustrates that, even if there are not any weaker edges to be found, the backtracking process can produce the best regularized solution. Fig. 5.19 (a) presents a contour with a large opening. Figs. 5.19 (b) and (c) show the first two equilibrium states, with one over-inflated and one over-deflated. After several more cycles, the balloon produces a better solution (Fig. 5.19 (d)), which is neither over-inflated nor over-deflated, because of the reduced t_{edge}.

The next several experiments use various real images to test the Actively Searching Contour Models. Figs. 5.20, 5.21, 5.22, and 5.23 are a group of ultrasound loin images taken from live pigs. The ASCMs are used to outline the contour of the loin object, and a subsequent step is conducted to estimate the size and weight. Each of the figures, except for Fig. 5.23, shows the original image, a saliency map produced by the pre-processing module, and the first and last equilibrium states of the ASCMs.

Figs. 5.20 and 5.21 demonstrate the ASCMs' ability to converge to contours with various gaps. By invoking the backtracking process, the final results provide improved regularized solutions for large gaps. This property is also illustrated by Fig. 5.22. Fig. 5.23 shows that ASCMs can skip less strong curves inside the loin object, and converge to the real boundary.

Fig. 5.24 using the example of a cup image. Results presented are produced by an adaptive balloon, as well as the traditional balloon model. It should be noted that the balloon using a constant force overruns the weak part of the contour and converges to the outer rim of the cup.

Fig. 5.25 is a CT medical image example that demonstrates the versatility of the ASCMs. By changing the direction of the initial inflating force, the balloon can either expand or shrink to search for the contour. For the expanding balloon, an appropriate inflating force is chosen so that the balloon converges to the contour shown in Fig. 5.25 (f), instead of those less strong curves inside the contour.

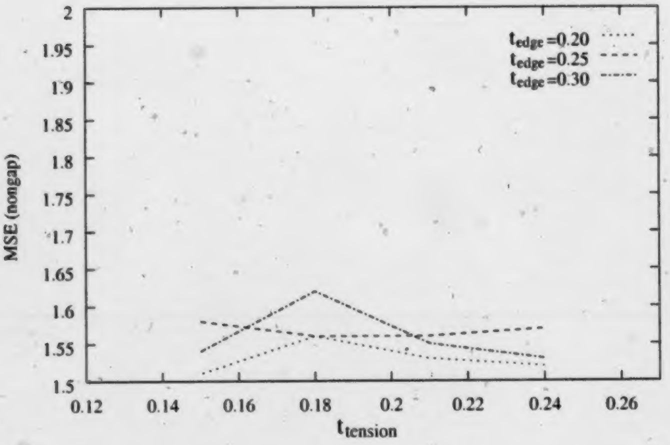
5.6 Conclusions

In this chapter, we first present a pre-processing method that produces reliable potential fields for snakes, from noisy images. The second part describes an adaptive inflating force which is easier to control and has fewer side effects. Finally, the methods introduced in Chapters 2 to 5 are put together in a framework to form the Actively Searching Contour Models.

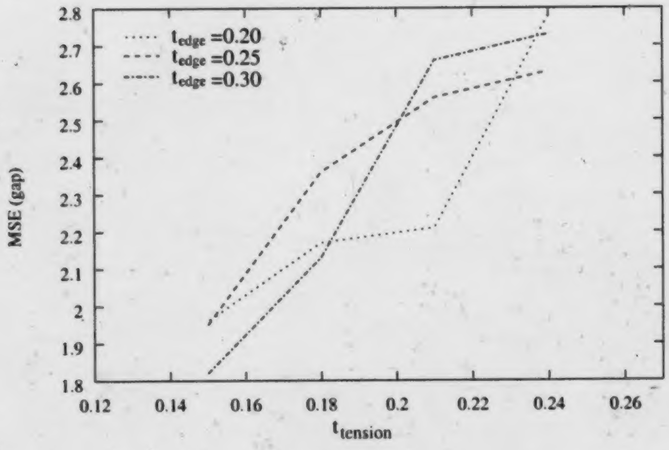
The pre-processing method is based on the Saliency Map method proposed by Sha'ashua *et al.*, and is quite robust against random noise. The SM method uses an iterative process to gather local information. Edges from object boundaries are reinforced using the information gathered, and noise edges are substantially reduced. A second step that performs hysteresis and non-maximum suppression is also introduced. Results show that the pre-processing method is effective for ultrasound images, and other images with random noise.

The adaptive inflating force is designed to reduce the side-effects resulting from a constant inflating force. By computing an inflating force locally for each snaxel, a snaxel is not subjected to an excessively large inflating force when the underlying image provides no image forces. Moreover, an adaptive inflating force is more accurate and controllable when two parameters are adopted, one used to control the strength of edges that a snake is allowed to converge to, and the other to control the inflation level of a balloon. A backtracking process is also introduced to search for any skipped less strong edges.

The methods introduced in this chapter are combined with those from previous chapters. The resulting framework is aimed at controlled accurate search. The advantages of the combined framework are threefold: (1) it can perform contour completion; (2) it can overcome edges below a given threshold; and (3) it allows for ease in choosing parameters. A diagram is provided to illustrate when and how to take advantage of all the separate modules to achieve the best performance. Finally, experimental results are provided to illustrate the advantages of adaptive balloons and the combined ASCMs.

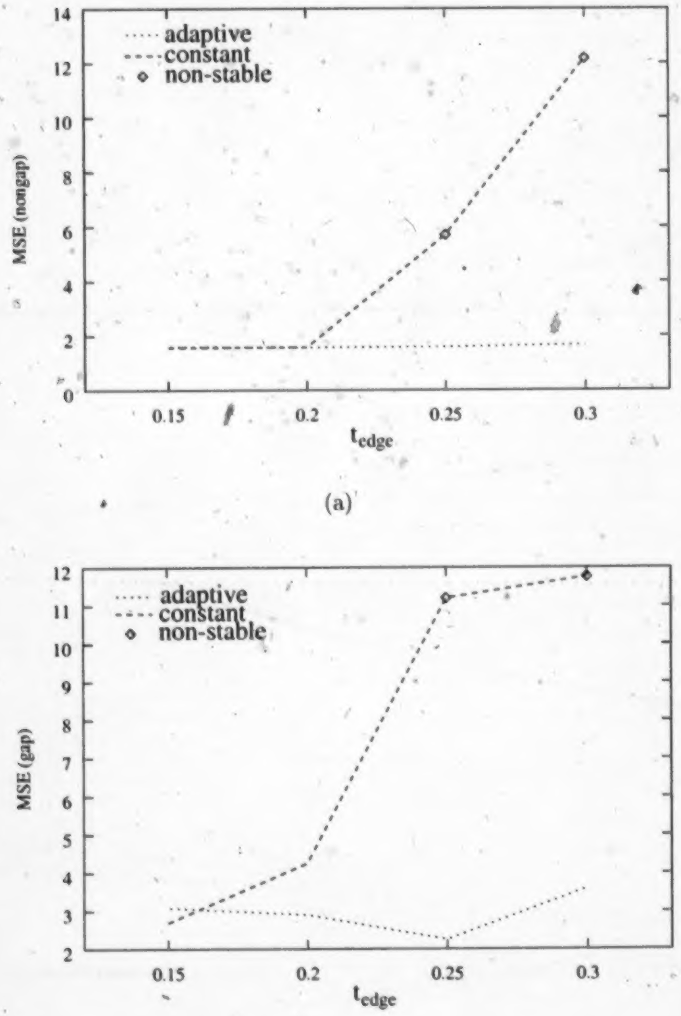


(a)



(b)

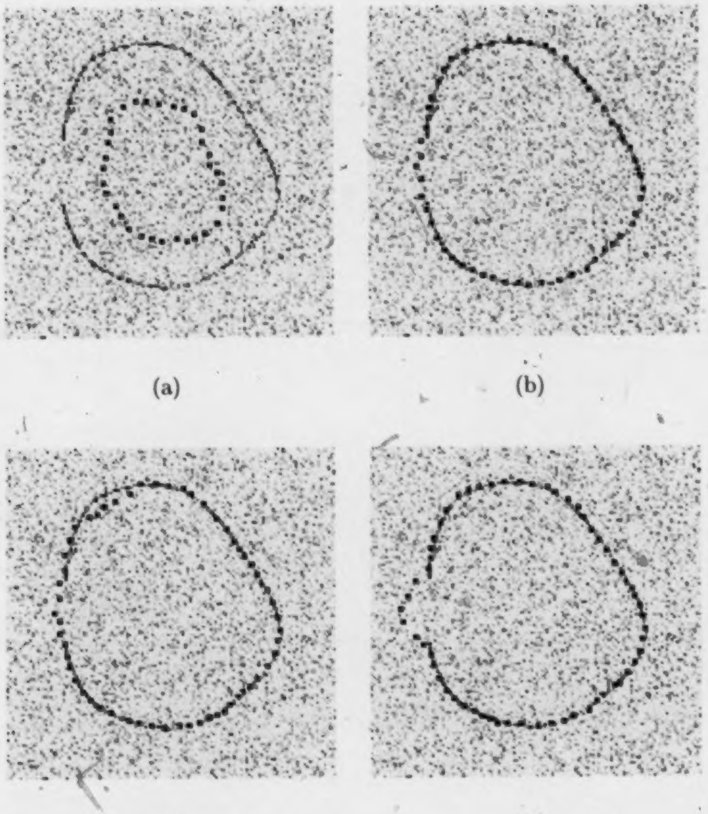
Figure 5.12: Testing the effect of t_{edge} and $t_{tension}$. The test image is a circle (radius 50) with four gaps of size 30.



(a)

(b)

Figure 5.13: Comparing an adaptive balloon with the traditional balloon. The parameter, $t_{tension}$, for the adaptive balloon is set to 0.15.



(a) (b)

(c) (d)

Figure 5.14: An experiment on parameter sensitivity: (a) a contour with a gap; (b) the result using the adaptive balloon; (c) and (d) results of the traditional balloon with different inflating forces. It should be noted that the balloon shown in (d) is not able to reach a stable state.

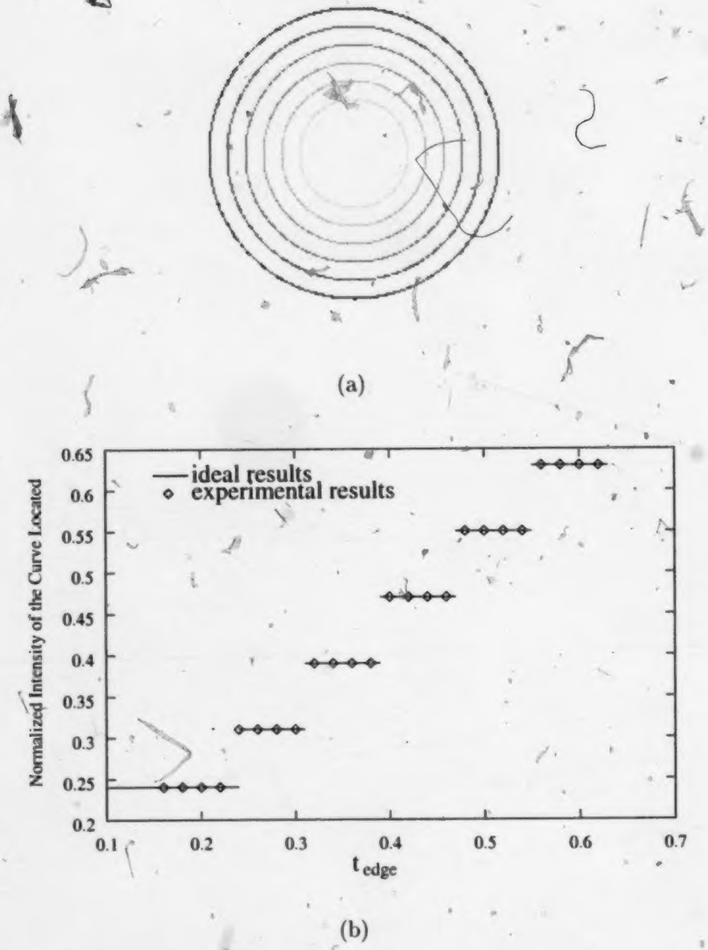
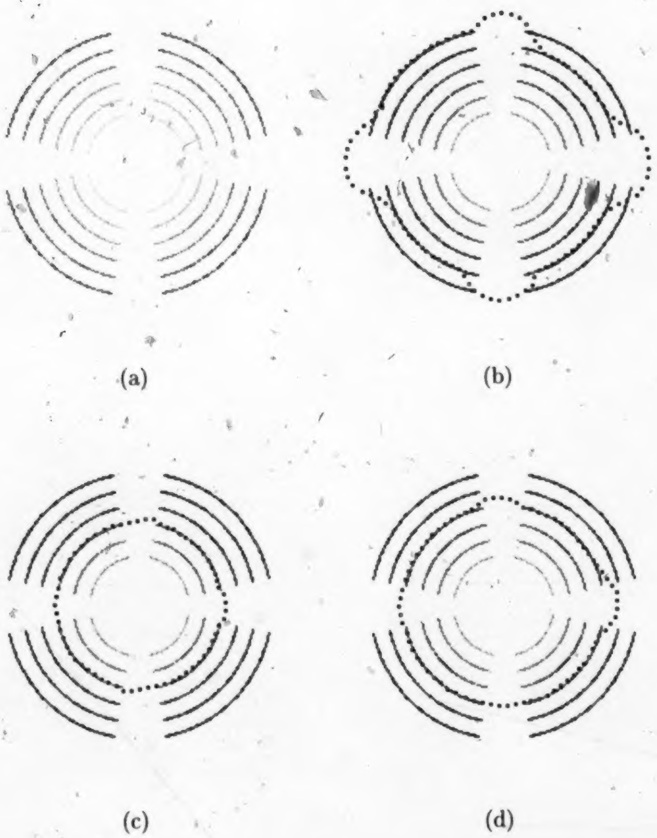


Figure 5.15: (a) a test image for accuracy testing. The third circle from inside has a radius of 50; (b) a plot of the extracted circles' intensity levels against the parameter t_{edge}.



(a) (b)

(c) (d)

Figure 5.16: (a) multiple concentric circles with gaps; (b) a result using the traditional balloon with a constant force of 0.35; (c) and (d) results of the adaptive balloon with the inflating force set to 0.35 and 0.41, respectively.

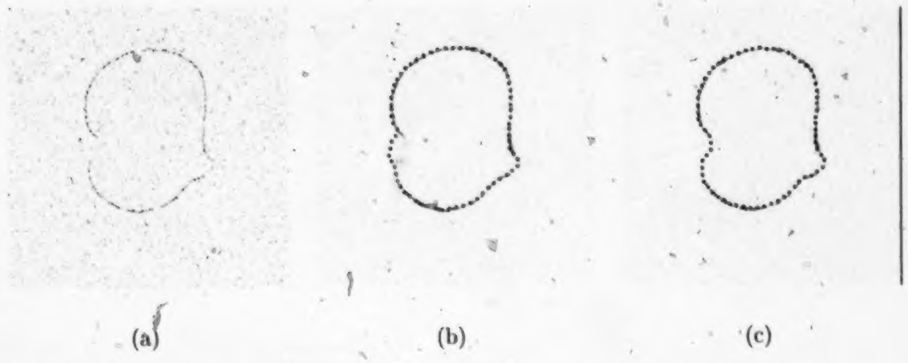


Figure 5.17: (a) a contour with some weak parts; (b) the first equilibrium state of an adaptive balloon; (c) the final result after several cycles of backtracking.

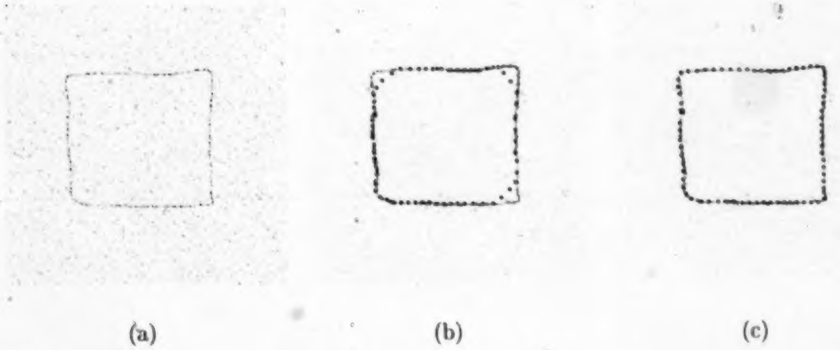
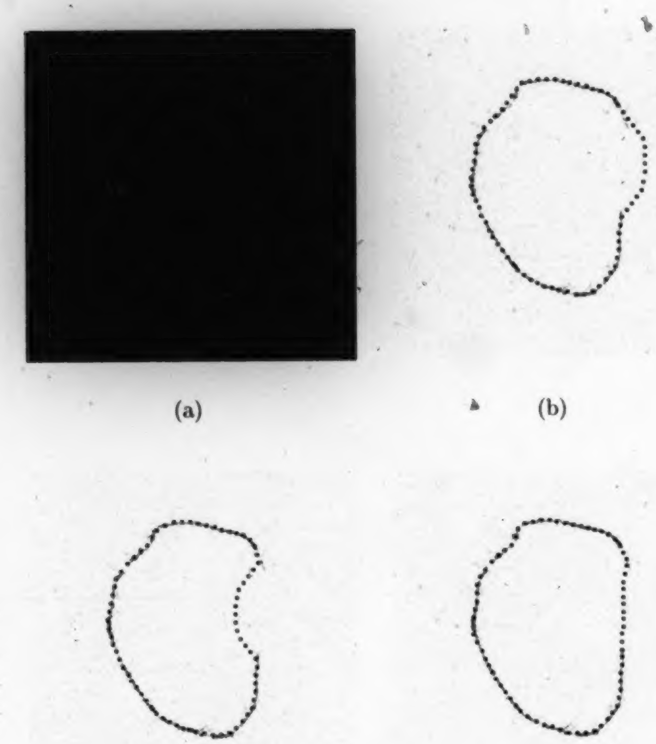


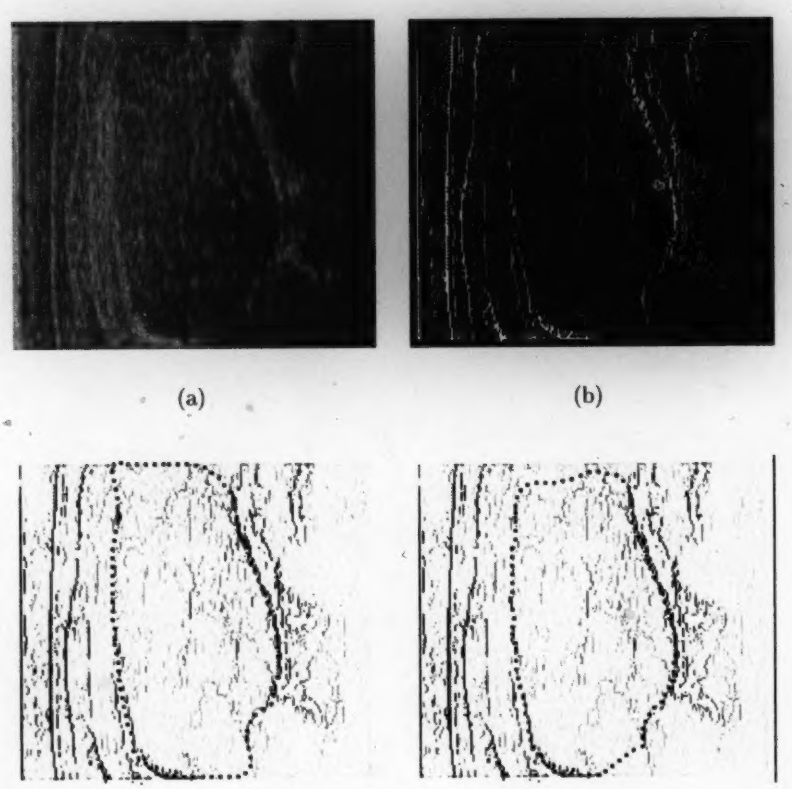
Figure 5.18: (a) a square with sharp corners; (b) a balloon with multi-scale and semi-rigid modules is not able to reach corners; (c) the balloon converges to the corners after disabling the two modules in the second sub-step.



(a) (b)

(c) (d)

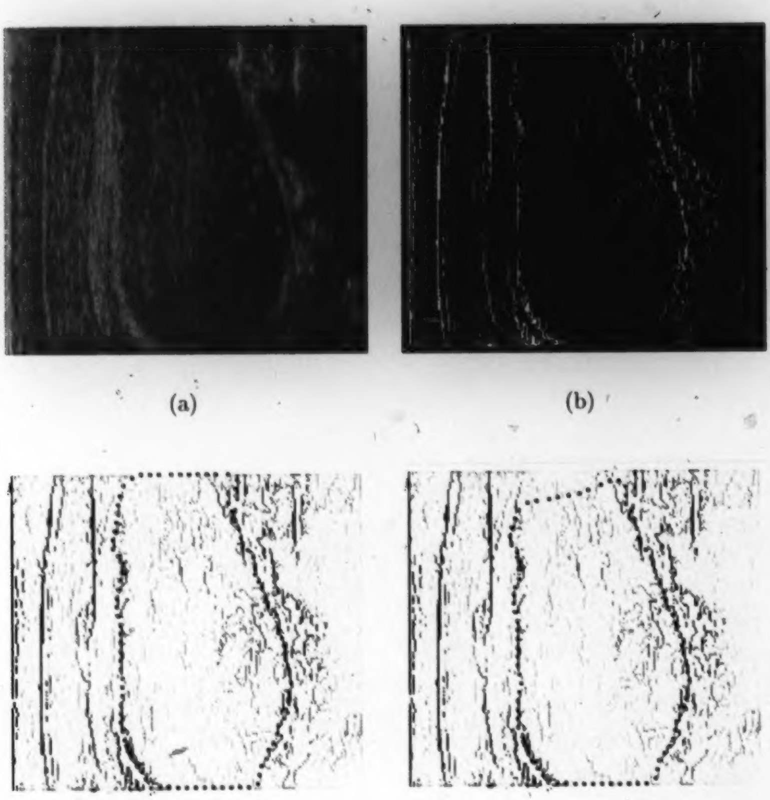
Figure 5.19: (a) a contour with a large opening; (b) and [illegible] show the first two equilibrium states of the backtracking process; (d) the final state, which is a better regularized solution.



(a) (b)

(c) (d)

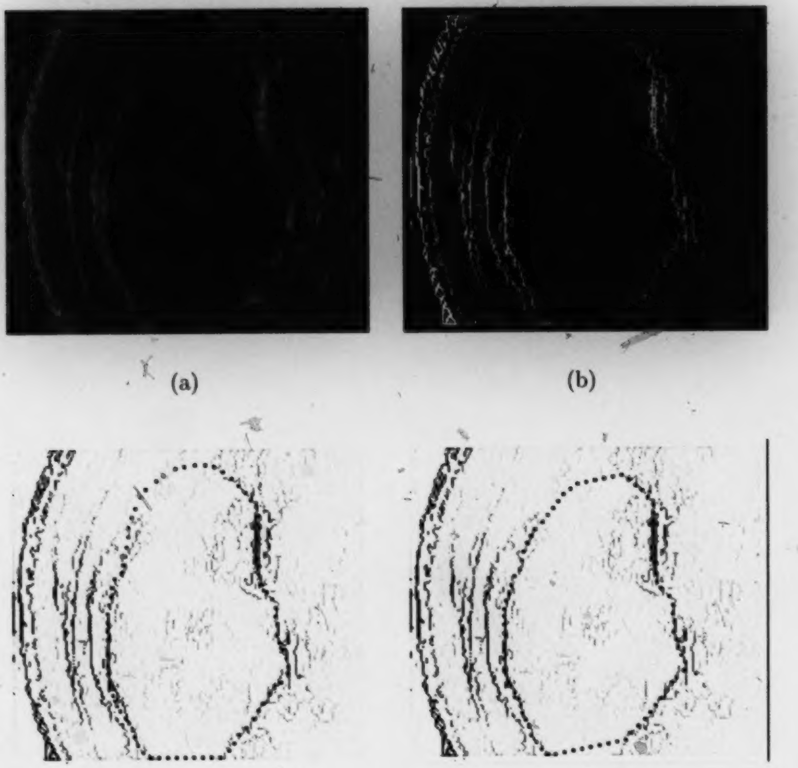
Figure 5.20: (a) a loin image; (b) its saliency map; (c) and (d) the first and final states.



(a) (b)

(c) (d)

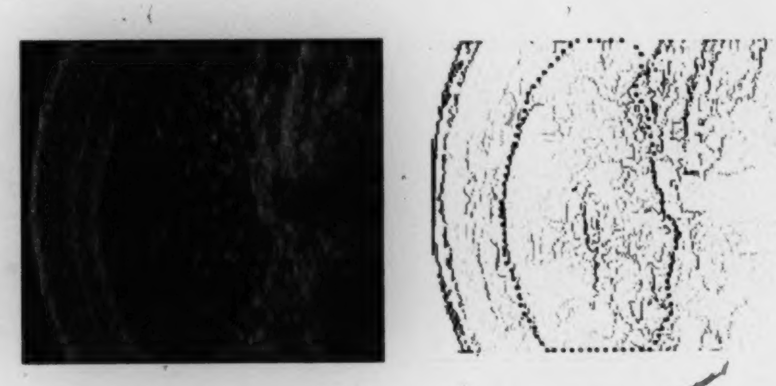
Figure 5.21: Another loin image example: (a)-(d) the loin image, its saliency map, and the first and last equilibrium states.



(a) (b)

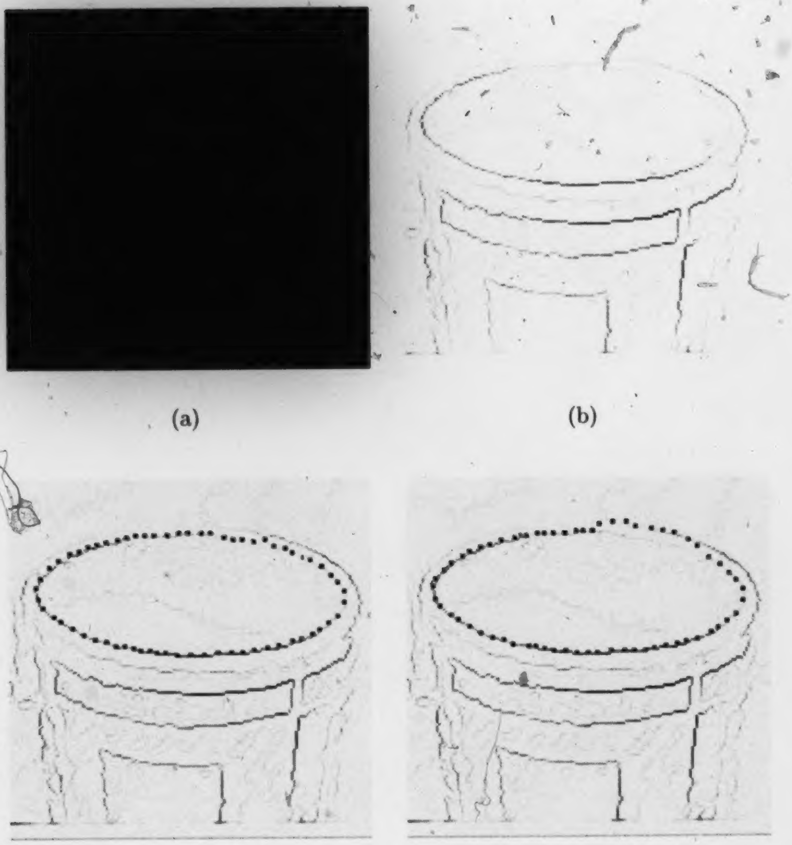
(c) (d)

Figure 5.22: The third loin image example: (a)-(d) the loin image, its saliency map, and the first and last states of a balloon.



(a) (b)

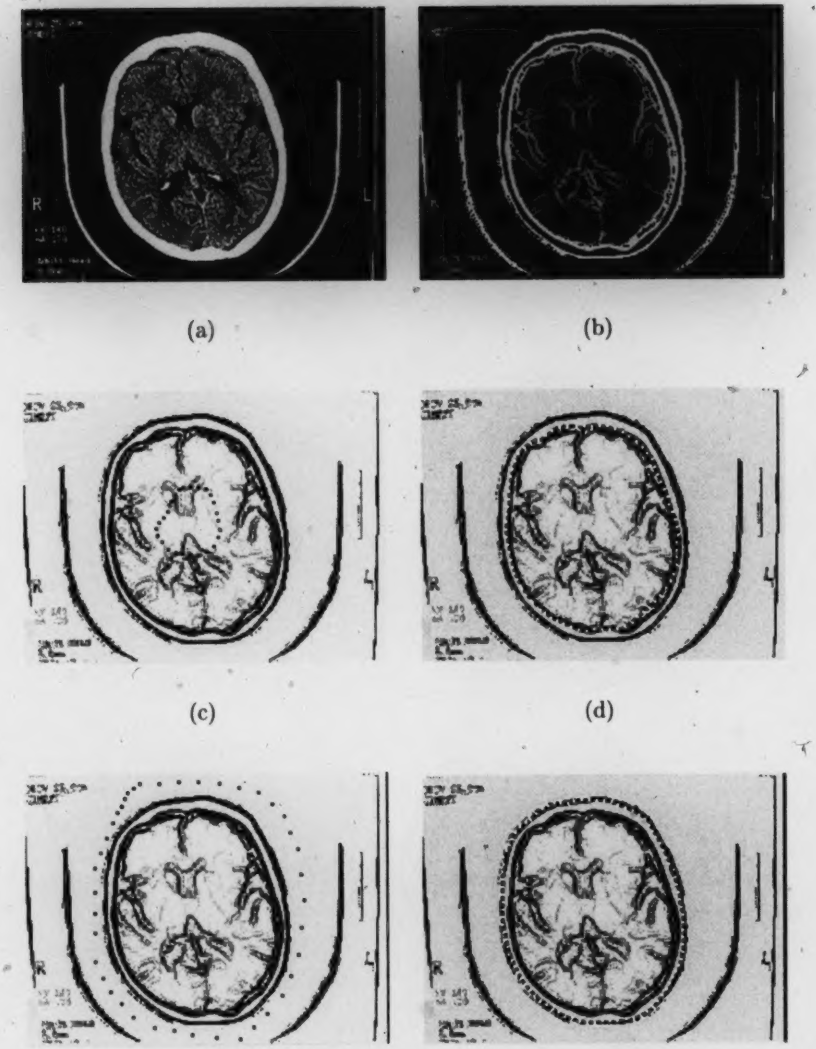
Figure 5.23: The fourth loin image example: (a) the loin image; (b) the contour extracted.



(a) (b)

(c) (d)

Figure 5.24: A cup example: (a) the cup image; (b) its saliency map; (c) the adaptive result; (d) the result produced by the traditional balloon.



(a) (b)

(c) (d)

(e) (f)

Figure 5.25: A CT medical image example: (a) the CT image; (b) its saliency map; (c) and (d) an initialization for a shrinking snake, and the corresponding result; (e) and (f) an initialization for an expanding balloon, and the result produced.

Chapter 6

Shape-Model Guided Search

In this chapter, we present a method to combine a grammatical model that encodes *a priori* shape information with the ziplock snakes presented by Neuenschwander *et al.* A competing mechanism is adopted to take advantage of the shape models without inducing excessive computation. The resulting model-based ziplock snakes have many advantages over the original model: they can accurately locate contour features, produce more refined results, and deal with multiple contours, missing image cues and noise. This model-based ziplock snake is part of our Actively Searching Contour Models (ASCMs) although it is not based on balloons. However, it shares the idea of active search with the models introduced in the previous chapters and requires a trivial initialization as well.

6.1 Introduction

By combining image information and smoothness constraints, Active Contour Models, or snakes, are effective and robust in contour extraction. In most papers on snakes, an initialization close to the desired contour is assumed to be provided, a requirement which is inappropriate in many cases. The ziplock snake model [38] presented by Neuenschwander *et al.*, however, needs only two user-supplied endpoints. The optimization process for a ziplock snake starts from the two endpoints and progresses towards the center of the snake. During the process, the image potential is progressively turned on to clamp the two ends of the snake on to an image contour. The ziplock snake model reduces the initialization need for a snake to merely specifying two endpoints. However, a ziplock snake is easily confused by an image potential with multiple contours due to the lack of an inherent global model.

To take advantage of *a priori* information in contour extraction, Olstad and Torp proposed incorporating a grammatical model into snakes [39]. Regular expressions are used to describe contour models. For example, a description for the square in Fig. 6.1 (a) is $(LL^*R)^*$, where L stands for a short straight line and R a 90° angle. In their experiments, for this particular example, 35 evenly spaced radial rays were chosen with 40 candidate points on each of them (Fig. 6.1 (b): only a few rays and points are shown for typographical purposes.) A candidate solution is a polygonal

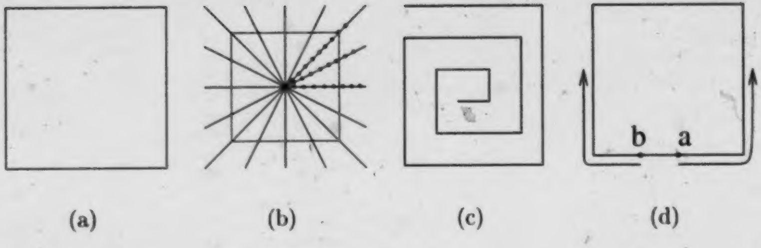


Figure 6.1: (a) a square; (b) rays and candidate points for the Olstad method; (c) a difficult case for choosing candidate points; (d) two user-selected starting points for a ziplock snake.

curve that connects a series of points with one from each ray. To apply the *a priori* information, each point is in turn assigned every possible symbol from the above regular expression. Any discrepancies between the actual curvature and the one associated with the symbol produce a penalty in the format of an energy term. The minimization is achieved by using the time-delayed discrete dynamic programming [3] method. The following problems with their method were noted: (1) A new set of candidate points needs to be specified for every new image. The number and positions of these points are not easy to decide (For an example, see Fig. 6.1 (c).) (2) The amount of computation is proportional to the square of the number of points. (3) The resulting grammatical representation is not necessarily a valid sentence of the given regular expression. One thing to emphasize is that the neighborhood used by their method is all the points on a specific ray. This number is usually much larger than the traditional neighborhood, which is a 3x3 grid. In essence, Olstad *et al.*'s method searches a large space[1] for a combination that best fits the given regular expression. The computation is prohibitive if the images are large.

In this chapter, we devise a method that combines a grammatical model with ziplock snakes. The resulting model-guided ziplock snakes are able to differentiate model-conforming contours from those not conforming, and thus the computation is moderate. In the remaining part of this chapter, Section 2 discusses how to encode *a priori* information and Section 3 presents how ziplock snakes work. How to integrate them is proposed in Section 4 and the results are reported in Section 5.

6.2 Encoding *a priori* Information

Grammars, a powerful tool to describe shapes, have been widely used in Syntactic Pattern Recognition. In this paper, regular expressions are used to encode shape models. An example will be shown to illustrate the basic concepts. Readers are referred to [39] and references therein for a detailed account of the grammatical theory.

Since ziplock snakes start from two user-supplied endpoints, the encoding should start from those points as well. For the square shown in Fig. 6.1 (d), if positions a and b are selected by a user, then $L^*RL^*RL^*RL^*R$[2] fully describes the counter-clockwise contour starting from a. An appropriate regular expression can also be given for the clockwise contour starting from b. Fig. 6.2 shows a nondeterministic finite-state machine that could be used to match patterns to this regular expression.

---

[1] It is in fact the whole space that is occupied by the contour to be extracted.

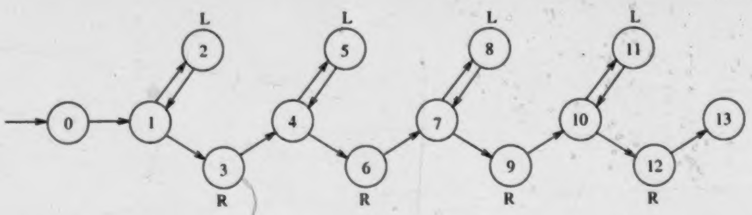


Figure 6.2: A nondeterministic finite-state machine for the regular expression that describes a rectangle.

A state table is generated for every nondeterministic finite-state machine. Table 6.1 shows the corresponding state table for the above example. For each state, there are one or two next states associated with it, depending on whether it is a terminal state or not. For a terminal state, a symbol is also associated with it. To further represent the expected length of certain lines or arcs, we add an average length and a deviation value for some terminal states (such as states 2 and 5) as attributes to the grammar.

6.3 How Ziplock Snakes Work

A ziplock snake consists of two parts: an *active* part and a *passive* part (see Fig. 6.3). The two parts are separated by moving *force boundaries*, and the active part is further divided into two segments, indicated as *head* and *tail* respectively. The initial positions of the head and tail segments are specified by an operator and/or a pre-processing module. The passive part is then generated by an interpolating method, such as the Bezier's curve method, using the force boundaries as input. Unlike the procedure for a traditional snake, the image potential is turned on only for the active part of a ziplock snake. Starting from two short pieces, the active part is iteratively optimized, and the force boundaries are progressively moved towards the center of the snake. Each time that the force boundaries are moved, the passive part is re-interpolated using the position and direction of the end vertices of the two active segments.

Compared with traditional snake models, ziplock snakes need far less initialization effort and are less affected by the shrinking effect from the internal energy term. Moreover, the computation process is more robust because the small excursion assumption is always satisfied [38], *i.e.* the active part whose energy is minimized is

---

[2] $(L^*R)^*$ is a more compact regular expression that generates approximately the same set of sentences. The longer expression is chosen because it can accommodate attributes for each symbol. For example, different attributes can be associated with each of the four $L's$ in the longer expression.

State	Symbol	Next1	Next2	μ	σ
0	-	1	-	-	-
1	-	2	3	-	-
2	L	1	-	5	2
3	R	4	-	-	-
4	-	5	6	-	-
5	L	4	-	20	5
6	R	7	-	-	-
7	-	8	9	-	-
8	L	7	-	20	5
9	R	10	-	-	-
10	-	11	12	-	-
11	L	10	-	10	4
12	R	13	-	-	-
13	-	-	-	-	-

Table 6.1: The corresponding state table for the automaton shown in Fig. 6.2. μ and σ are the expected length and deviation of a certain graphical element such as an arc or a straight line denoted by the corresponding symbol.

always quite close to the contour being extracted. The small excursion assumption is necessary for the underlying mathematical theory of the deformable method, and this is why a close initialization has to be provided for the traditional snake models.

As stated in Section 1, ziplock snakes are easily confused by multiple contours or missing contour parts due to the lack of a global model. Fig. 6.4 shows two difficult cases for ziplock snakes. For the image with multiple contours, a ziplock snake is not able to tell which direction is more appropriate, and the result depends largely on the shape and position of the passive part. For the contour with gaps, a ziplock snake would stop at the gaps and find only part of the contour.

6.4 Combining Ziplock Snakes with the Grammatical Model

Unlike the traditional snake models and balloons, a ziplock snake starts from two endpoints, thus making it possible for sequentially encoded shape model information to govern the searching process. To use the shape information with a balloon, the shape model has to be aligned with the balloon before each round of energy minimization. If only one alignment is used, the error involved in this process will limit the performance of the shape model. This is the main reason why a ziplock snake is chosen instead of a balloon. Possible solutions to combining a shape model with a balloon are discussed in the Future Work section of Chapter 7.

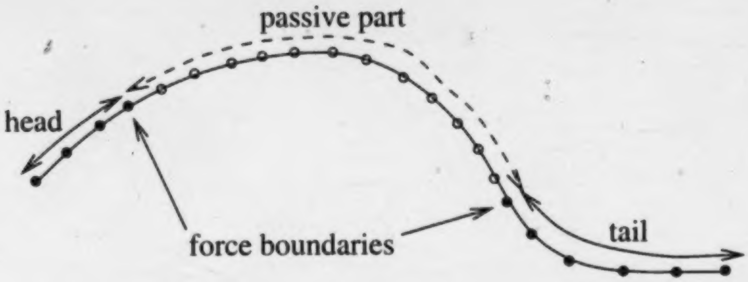


Figure 6.3: An illustration of a ziplock snake. The head and tail are the active parts that are under the influence of the image potential. The force boundaries are moved gradually to convert more of the passive part into active part.

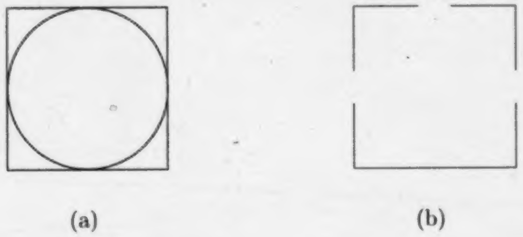

(a)
(b)

Figure 6.4: Difficult cases for ziplock snakes: (a) multiple contours crossing one another; (b) a single contour with gaps.

6.4.1 Competing Interpretations

Given the regular expression that describes a contour to be extracted, one can generate every possible sentence that can be used to interpret the two active segments. For example, for the counter-clockwise contour starting from a as shown in Fig. 6.1 (d), if the active segment is only 3 nodes long (excluding the first and last nodes), then all the possible valid sentences or interpretations are *LLL, LLR, LRR, LRL, RLL, RLR, RRR,* and *RRL.*

For this active segment, the best next-position[3] is computed for every node using each and every one of the interpretations as a constraint. The constraint is expressed by a potential energy that is dubbed the grammatical energy. For every snake node, the grammatical energy is computed based on the discrepancy between the curvature measured at this node and that given by the grammatical symbol assigned to it. The total energy for a specific interpretation is the sum of the grammatical energy of all nodes on the active segment. The best next-position derived using this interpretation is then recorded in a list, and then the process is repeated for the next interpretation. After all the computation is done, the next-position list is searched for the best (of all interpretations) move, *i.e.* the one with the smallest energy. The active segment is thus best described by the corresponding interpretation and is moved to the best next-position found.

---

[3] Considering a 3x3 neighborhood for every snake node [3].

The above method is computationally expensive. The amount of computation increases exponentially because the number of possible interpretations increases exponentially when the two active segments grow. Since many interpretations do not fit the underlying image information at all, the amount of computation can be reduced. We now present a way to trim unlikely interpretations while retaining the most reasonable ones. The idea is to keep the n most "promising" interpretations after each round of computation and to derive new interpretations only from them. "Promising" is measured by the energy value computed for an interpretation, and the smaller it is the better.

For an active segment of length l, the number of all possible interpretations with length from 1 to l is about $2^l - 1$ (the number of nodes on a complete binary tree), and the trimmed-down number is bound by nl. An example is shown in Fig. 6.5. One thing to note is that such a binary tree can be constructed for any contour. This is because for any interpretation, at most two new ones can be derived from it.

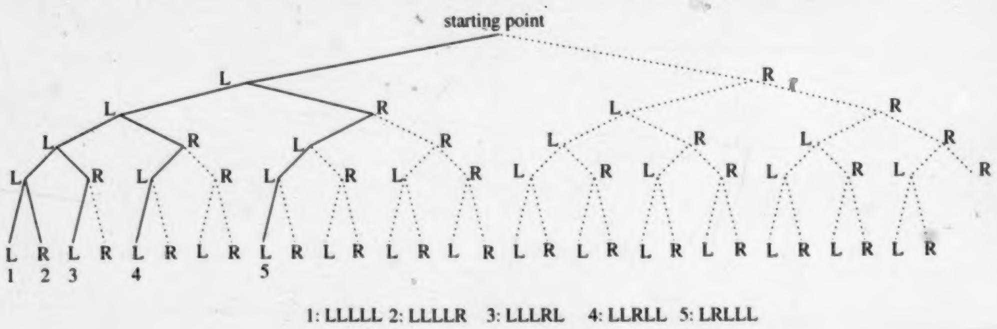


Figure 6.5: A binary tree listing all possible interpretations shorter than 6 for the regular expression $L^*RL^*RL^*RL^*R$. Solid lines indicate the paths to the five most possible interpretations.

Once a binary tree has been trimmed to a few branches, the energy minimization process is much faster. Each time, one of the interpretations is chosen as the best description of the active segment. Once the minimal energy has been reached, the force boundary of this active segment is moved one node closer to the center of the snake. To accommodate this new node, new interpretations are generated from those n old ones. Since each interpretation can have two derivations, $2n$ new interpretations are generated and n of those with smallest energies are retained.

The purpose of keeping n most probable interpretations instead of one is to tolerate errors and to introduce competition. Even if an interpretation is not considered as the best one at one point, there is still a chance that one of its derivations may be when new information is introduced.

In conclusion, by having the n best interpretations competing against one another, a ziplock snake is not only governed by a valid instantiation of a shape model, but also open to new interpretations when the snake grows. The complexity of this algorithm is bound by n times that of the original ziplock snakes. However, a model-based ziplock snake converges more quickly and needs fewer iterations.

6.4.2 Predicting the Next Step

After the two active segments are optimized, the passive part is re-interpolated based on the force boundaries (marked in Fig. 6.3). The position and tangent of these two points are the input to an interpolating method such as Bezier's method. As introduced in Neuenschwander *et al.*'s paper, the interpolating process provides a regularized solution for the passive part. Moreover, it allows the snake to grow or shrink in a natural way, thus avoiding the shrinking problem of the original snake models.

After the interpolation, each of the force boundaries is moved one node closer towards each other. In other words, the interpolating process predicts a next step for each of the active segments. Now that a grammatical model has been incorporated and a list of n most reasonable interpretations is available, more information can be fed to the interpolating method to produce a better next step. Since the regular expression encodes shape information, the next symbol of an interpretation contains *a priori* information. On the other hand, image information is available to compute a likelihood value for each and every next symbol generated. Combining these two sources, *posteriori* probability values can be computed for every next symbol using Bayes's rule.

Since there are n possible interpretations and each of them can produce one or two symbols, about $2n$ next steps are generated. We now use m to represent the actual number of symbols generated. Each next symbol contains curvature and direction information for a possible next step, and, therefore, a likelihood value can be computed by matching this information to the image content. A *posteriori* probability value for every next symbol is then computed using the following formula:

$$\begin{aligned} P(ns_i|l) &= \frac{P(ns_i)P(l|ns_i)}{\sum_{i=1}^{m} P(ns_i)P(l|ns_i)} && (6.1) \\ &\propto P(ns_i)P(l|ns_i) && (6.2) \end{aligned}$$

where ns_i is the ith next symbol and l stands for the image content. The denominator is the same for every next symbol and is removed from further computation.

To estimate $P(ns_i)$, we make use of two categories of information: (1) The energy value computed for an active segment using the interpretation from which ns_i is derived as a constraint. (2) A probability value computed for ns_i based on expected length of line or arc that the current symbol stands for. This is to take advantage of the attributes added to the regular expression. For example, if L and R are both valid next symbols of the interpretation $LLLLL$, and the length of the active segment is approximately the same as the attribute μ given for current symbol L, then R is the more probable next symbol. Formally, $P(L|\mu(cs), \sigma(cs))$ is computed using:

$$P(L) = \int_{-\infty}^{L} \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)/(2\sigma^2)} dx \qquad (6.3)$$

$$P(L|\mu(cs),\sigma(cs)) = \begin{cases} P(L) & \text{if } ns_i \neq cs; \\ 1.0 - P(L) & \text{otherwise.} \end{cases} \tag{6.4}$$

where $\mu(cs)$ and $\sigma(cs)$ are the mean and standard deviation of the expected length distribution for the current symbol cs. The length distribution is assumed to be Gaussian. A sample plot of $P(L)$ and $1.0 - P(L)$ is shown in Fig. 6.6 for $\mu = 100$ and $\sigma = 20$.

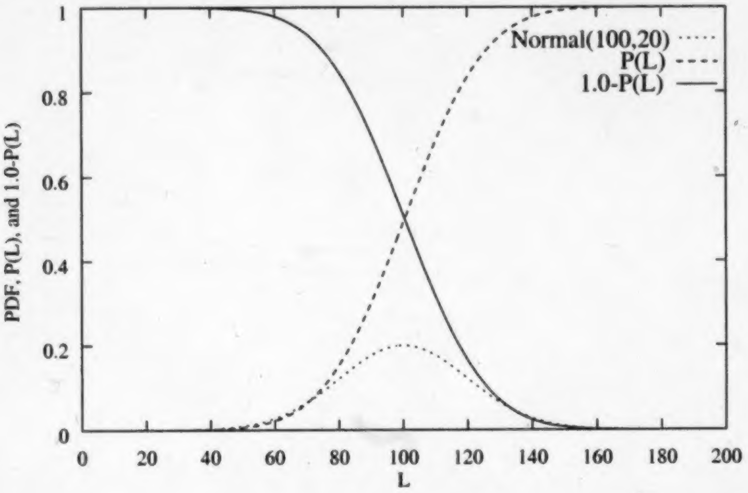


Figure 6.6: A plot of a Gaussian distribution with $\mu = 100$ and $\sigma = 20$. The corresponding $P(L)$ and $1.0 - P(L)$ are also shown.

Each of the next symbols is then assigned a probability value:

$$P(ns_i) = P(E(I))P(L|\mu(cs),\sigma(cs)), \tag{6.5}$$

where $P(E(I))$ is a probability value given the minimal energy $E(I))$ computed for interpretation I during the last energy minimization step.

The next step is to estimate $P(l|ns_i)$ which is a likelihood value for next symbol ns_i on the image content l. To estimate this value, all curves emanating from the force boundaries are first traced. This task can be efficiently done by using the saliency map method reported in Chapter 4 and needs to be done only once at the pre-processing stage. Let k stand for the number of curves traced. Once all the possible curves are traced, a compatibility value is computed for every combination of a next symbol and a traced curve. For every next symbol, k probability values are computed and the largest one is chosen as $P(l|ns_i)$.

As an example, Fig. 6.7 (a) shows the current state of a ziplock snake and (b) shows two possible next steps for the right side active segment. Fig. 6.7 (c) displays all three curves traced from the corresponding force boundary. To compute a compatibility value for the No. 1 next step and the No. 1 traced curve, we chose to use the area encircled by them (Fig. 6.7 (d)). The size of the area indicates the distance between the two items. Precisely, $P(l|ns_i)$ is computed using:

$$\begin{aligned} P(l_i|ns_i) &= S(l_i) * exp(-\rho A(l_i, ns_i)); && (6.6) \\ P(l|ns_i) &= \max_{i=1}^{k} P(l_i|ns_i), && (6.7) \end{aligned}$$

where function $A(l_i, ns_i)$ is the normalized area encircled by l_i and ns_i within a predefined radius. Function $S(l_i)$ is the normalized average edge magnitude on the traced curve l_i.

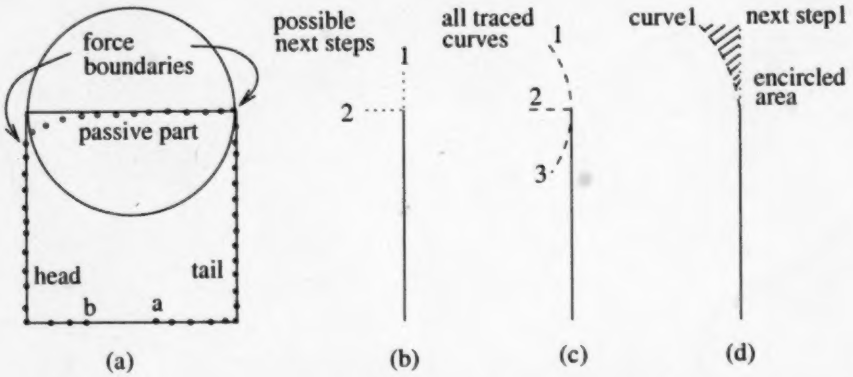


Figure 6.7: (a) an intermediate state of a ziplock snake; (b) all possible next steps for the right-side active segment; (c) all traced curves from the corresponding force boundary; (d) compatibility is mapped through an exponential function of the shaded area.

6.4.3 Further Reducing Computation

Since only the n most promising interpretations are retained after each round of optimization, they tend to converge to one another. By converging, we mean that the n interpretations share the leading part. For example, {*LLLLL, LLLLR, LLLRR, LLLRL*} is the group of interpretations our algorithm produces for the square in Fig. 6.1 (d) at some point. The common leading part is *LLL*. All others are eliminated because *LLL* best fits the underlying image information, and interpretations derived from it have larger probability values.

Since this behavior is expected in a typical execution, computation can be reduced by stopping the optimization for the leading common part. The different part, which is where the competition is focussed on, is of fairly fixed length most of time.

6.5 Implementation and Experiments

The algorithm is implemented using the time-delayed discrete dynamic programming [3] method. Grammatical energies are mapped through an exponential function:

$$E_{grammar} = \begin{cases} 1 - \exp(-\beta(ExpAng - x)) & x \leq ExpAng; \\ 1 - \exp(-\beta(x - ExpAng)) & x > ExpAng. \end{cases}$$

where $ExpAng$ is the expected angle (from the grammar) and x is the measured one. The parameter β is used to adjust the steepness of the energy function and is set to 0.5 in our experiments.

The advantages of our algorithm are illustrated in the following experiments. The first experiment demonstrates its ability to accurately locate contour features, thanks to its underlying competing mechanism. For example, if given a regular expression L^*CL^* (symbol C has an expected angle of 135°) and an input image (Fig. 6.8 (a)), then an intermediate state is as illustrated in Fig. 6.8 (b). At this point, the 4 most promising interpretations are $\{LLC,\ LLL,\ LCL,\ CLL\}$[4], all of which show that the competition is now focusing on the position of feature C. After more iterations (Fig. 6.8 (c)), the list becomes $\{LLLLLLC,\ LLLLLCL,\ LLLLLLL,\ LLLLCLL\}$, which means the snake has found a better match for C and the competition has shifted to this new area.

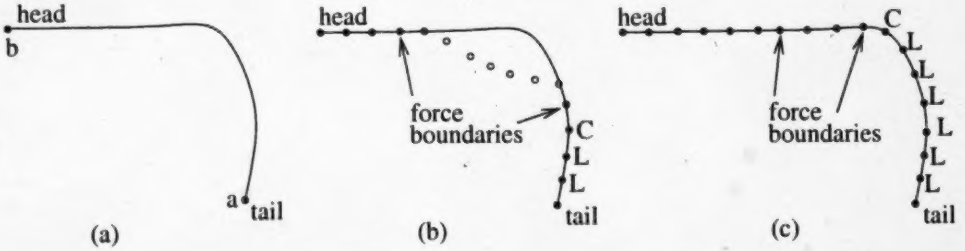


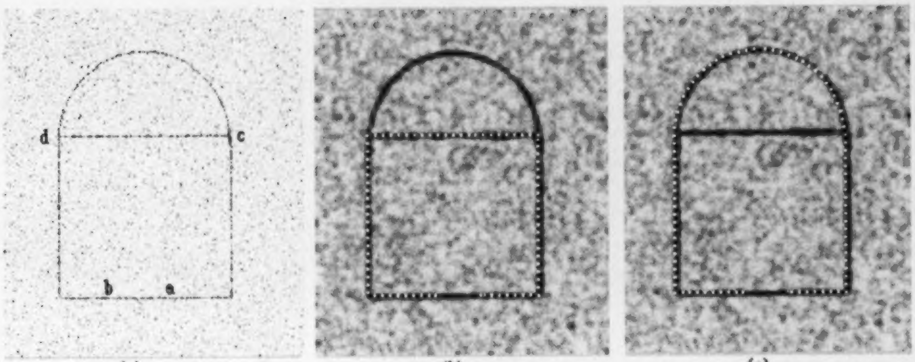
Figure 6.8: (a) a contour; (b) an intermediate state; (c) after some more iterations.

The second experiment illustrates the ability of the algorithm to deal with multiple contours. Fig. 6.9 (a) shows an image with two overlapping contours, where a and b are the two user-initialized points. The original ziplock snakes fail because at positions c and d they have no knowledge as to which contour to choose. With our algorithm, necessary information is effectively encoded using the regular expressions. If given the description for a square and necessary information on the length of the side lines, the square is extracted (Fig.6.9 (b)). Fig. 6.9 (c) shows the arch extracted when the description is changed to that of an arch.

The third experiment tests the robustness of our algorithm. As shown in Fig. 6.10 (a), a model-based ziplock snake is able to continue searching and find the right result even when image cues are missing temporarily (Fig. 6.10 (b)).

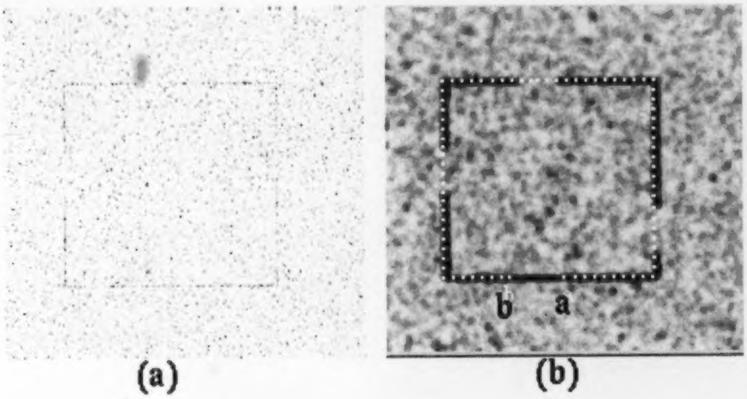
---

[4] The content of this list depends on the parameters chosen. Symbols are assigned to all snake nodes excluding the first and last ones.



(a) (b) (c)

Figure 6.9: (a) multiple contours; (b) the square is extracted when given a description for a square; (c) the arch extracted when the shape model is for an arch.



(a) (b)

Figure 6.10: (a) a square with some gaps; (b) the square extracted successfully.

The robustness of the model-based ziplock snake is also illustrated by the following two experiments. The test image (Fig. 6.11 (a)) is again a square with missing parts, similar to the one used in the last experiment. However, the square in Figs. 6.11 (a) and 6.12 (a) has a unit contrast, and the background is filled with Gaussian noise with zero mean and a standard deviation of 1.0 and 1.2 respectively. Figs. 6.11 (b) and 6.12 (b) show the saliency maps generated by the pre-processing module for curve tracing. Even with such a high level of noise, the squares are successfully extracted (Figs. 6.11 (e) and 6.12 (c)) with the help of a shape model. During one searching process, the ziplock snake deviates a little from the straight side line that it is supposed to follow (see Figs. 6.11 (c)-(d)). However, when it locates strong curves later, the deviation is corrected thanks to the shape model. In Figs. 6.11 (f) and 6.12 (d), darker areas indicate the part of image that is searched by the curve tracing procedure for possible next steps.

The last experiment shows the power of integrating attributes to the shape model.

Shown in Fig. 6.13 (a) is an image with several rectangles. Without any attributes, the regular expression to describe any of the rectangles is exactly the same, thus making it impossible to tell one from another. With our model-based ziplock snakes, we simply specify the expected length and deviation for the side lines of each rectangle. With this extra *a priori* input, the right course is selected after *posteriori* probability is computed for every possible next step.

6.6 Conclusions

In this chapter, a shape model-based ziplock snake is presented and is shown to have some excellent properties. The shape models are encoded with attributed regular expressions that are more powerful in describing shapes than the regular expressions used by Olstad *et al.* Attributes of a grammar are sometimes indispensable if the shapes to be extracted are structurally similar.

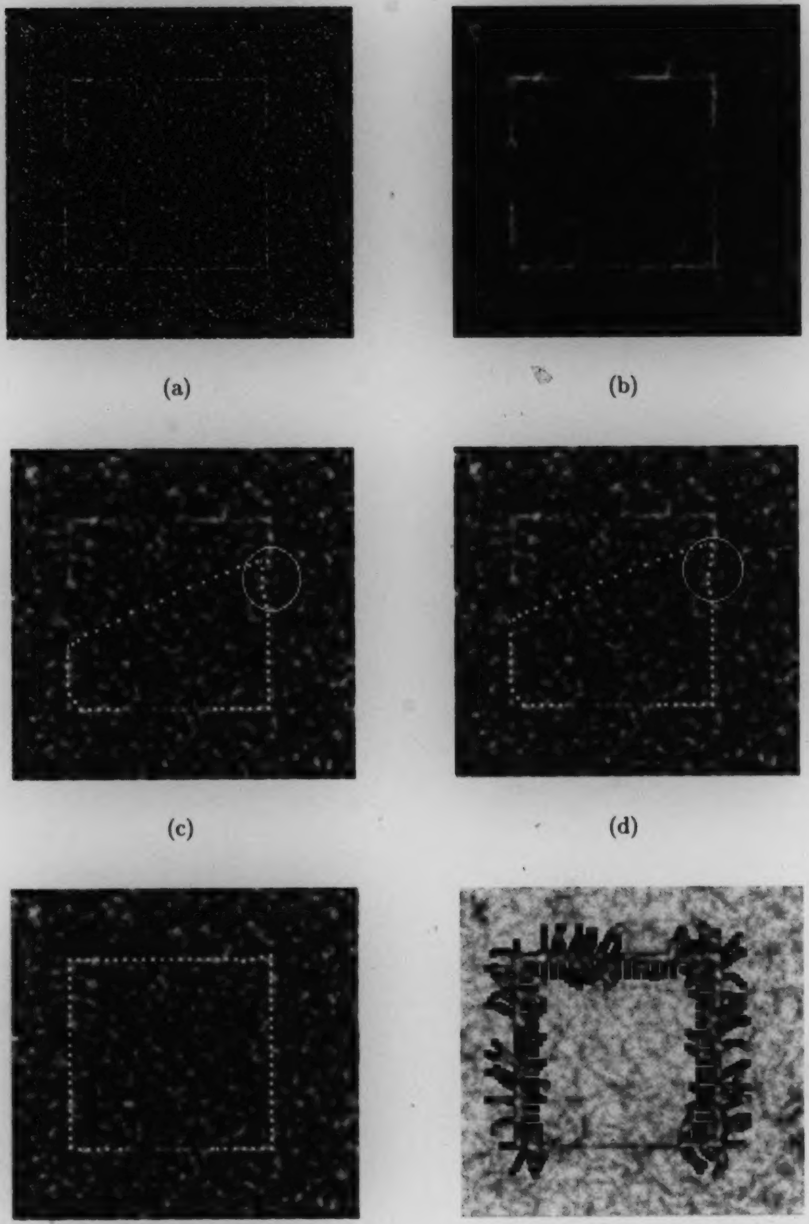
The shape models are then used to guide the searching process of a ziplock snake. The ziplock snake models are chosen over the balloon models because the shape models are sequentially encoded. To combine shape models with balloons, a difficult alignment problem has to be solved first. Similar to the balloons, the ziplock snakes need trivial initializations, too, and can be used in automatic image-processing applications.

To guide a ziplock snake's searching process, all valid sentences of the specified regular expression are generated step by step. Each of the symbols in a sentence is assigned to a snake node, and a grammatical energy is computed to reflect the discrepancies between the attributes of the symbol and the actual values computed for the corresponding snake node. By including the grammatical energy in the total energy formulation, a ziplock snake is influenced by the shape models.

To reduce the computation, only a few most reasonable sentences are retained every time a snake's energy is minimized. The sentences are designed to compete against each other, and this behavior helps a snake accurately locate contour features.

Each time the ziplock snake is expanded, a new node is added. With the shape information available, the position of the new node is predicted by combining *a priori* information and the image content. *Posteriori* probability values are computed for all the next steps generated from the retained sentences, and the largest one is chosen as the best prediction. By using such a sophisticated next-step predicting method, the model-based ziplock snake is able to differentiate between multiple contours and extract the right one.

With a grammatical model integrated, the new ziplock snake model can now handle multiple contours, noise, and missing image cues, all of which the original model cannot. Experiments are provided to illustrate all these features.

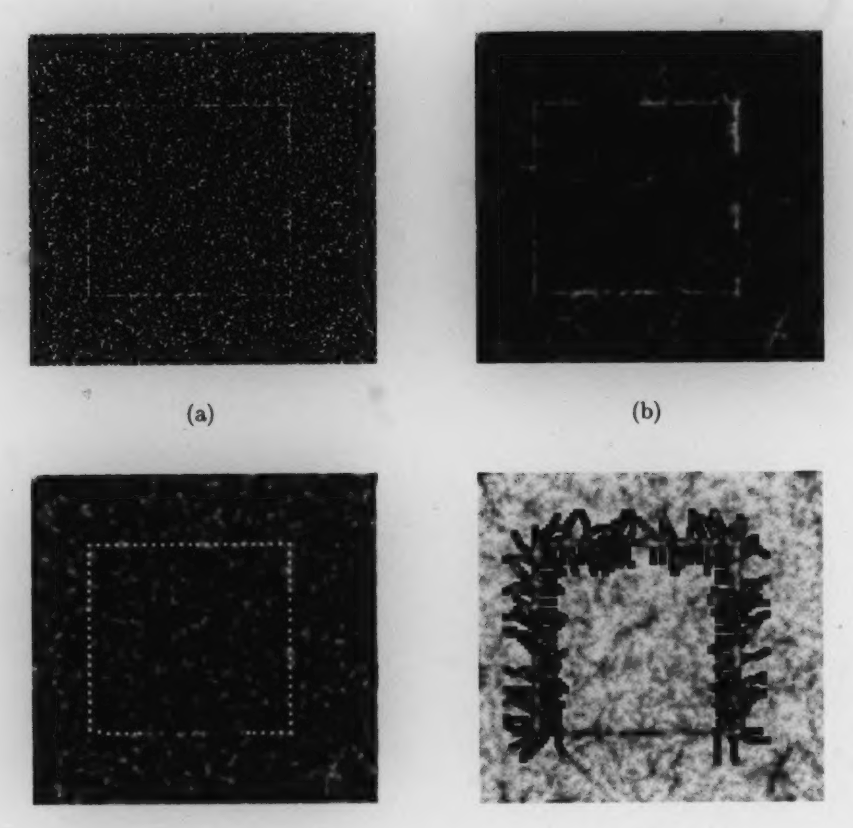


(a) (b)

(c) (d)

(e) (f)

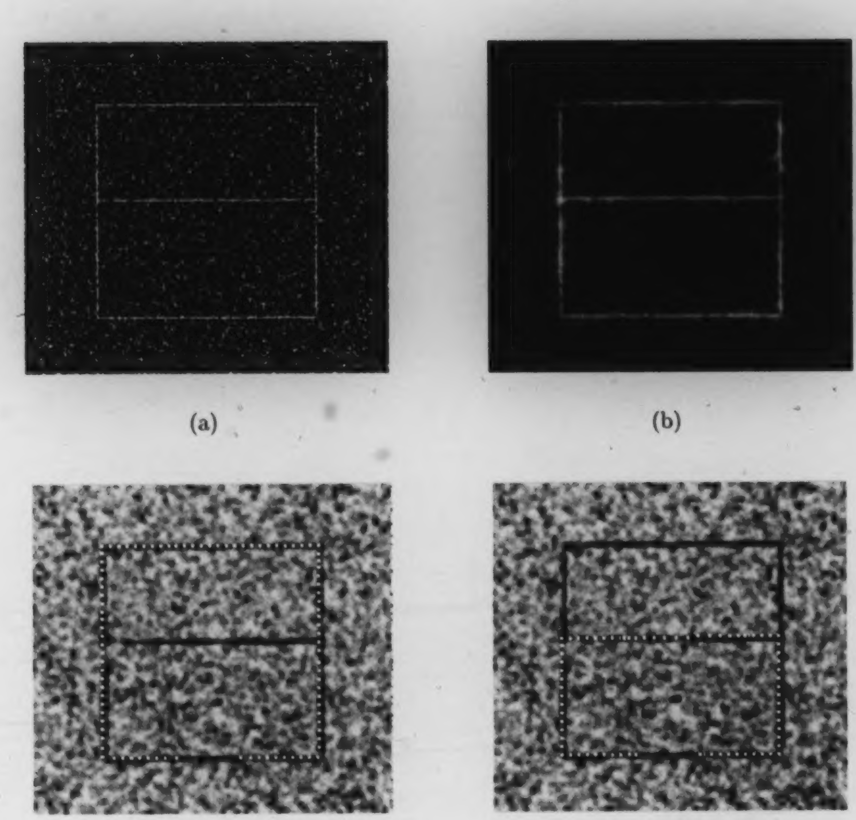
Figure 6.11: (a) a square with unit contrast, and corrupted by zero-mean Gaussian noise with a standard deviation of 1.0; (b) the corresponding saliency map generated for curve tracing. (c) the ziplock snake deviates from the straight side line when it is attracted by noise; (d) the deviation is corrected once the snake locates strong curves; (e) the final results; (f) the darker area indicates the part of the image that is searched by the curve-tracing method.



(a) (b)

(c) (d)

Figure 6.12: (a) the same image as in Fig. 6.11 (a) except the noise standard deviation is increased to 1.2; (b) the corresponding saliency map; (c) the final result; (d) the area that is searched for possible next steps.



(a) (b)

(c) (d)

Figure 6.13: (a) an image with several rectangles; (b) the corresponding saliency map; (c) the largest rectangle is extracted when the expected length and standard deviation of the side lines are set to 100 and 20 respectively; (d) the lower half of the largest rectangle is extracted when the expected length is lowered to 70. The initializations and other parameters are exactly the same.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

We have presented a series of methods that comprise the Actively Searching Contour Models (ASCMs) System for contour extraction. ASCMs are designed to actively search for desired contours, as opposed to the traditional Active Contour Models' behavior of being passively attracted to image features.

The first part of our work on ASCMs is based on Cohen *et al.*'s balloon models and is aimed at extracting objects with fuzzy boundaries. We had three objectives in mind when we set out to improve the balloon models: (1) a balloon should perform contour completion when needed; (2) it should be able to overcome local minima and accurately converge to desired boundary; and (3) it should allow for ease of parameter choosing. We began by examining problems with the energy minimization method and the way that snakes are modeled. Semi-rigid models were then proposed to solve the peeling problem that was observed. Subsequently, we proposed a multi-scale non-shrinking internal energy model, which is also unbiased towards expanding or contracting. The multi-scale part of the internal energy model provides a balloon with a controllable smoothness constraint, which is essential for contour completion.

We then proposed using image content to help a balloon searching for contours. The saliency map method was utilized to pre-process input images, and to trace curves that pass through a certain pixel. The curves traced were then used to guide a balloon's searching process. With this extra input, the improved balloon model was faster and more robust in many cases. The extra input also enables a balloon to reach high-curvature areas of a contour.

To accurately search for contours, we then proposed: (1) using the saliency map method, and subsequent steps, to transform the input image into a good potential field for balloons; (2) use of an adaptive inflating force that is able to make a balloon converge to edges with specified strength. All the above methods were then combined into one framework, using the backtracking strategy.

The second part of the ASCMs project is based on Neuenschwander's ziplock snake models, which share the idea of active search and also require trivial initializations. To make a ziplock snake more robust, a grammatical model that encodes *a* *priori* shape information is integrated into it. A competing mechanism is proposed to take advantage of the shape models, without inducing excessive computation. The resulting model-based ziplock snakes can accurately locate contour features; produce more refined results; and deal with multiple contours, missing image cues, and noise.

The content of this thesis was partially reported in various papers [53, 51, 34, 49, 52, 50] written by the author.

7.2 Future Work

In the second part of our ASCMs' work (Chapter 6), we chose to incorporate a grammatical model into the ziplock snakes instead of into the balloon models. As discussed in Chapter 6, this is because the regular expressions encode *a priori* information sequentially. A balloon has to be aligned with the shape model before each round of energy minimization. However, the advantages of using a shape model are limited if errors are introduced in the aligning process.

Despite various methods proposed in the literature, a balloon is still likely to be stuck in local minima if there are significant irrelevant edges in the expanding course of a balloon. Shape models are essential in getting the balloon out of local minima.

One viable way to use shape models with balloons is: (1) Whenever a balloon reaches equilibrium (meaning local minima or the global minimum), the shape model is used to assess the balloon by matching it with the strong parts of the balloon. Several good matchings are generated for further use. (2) a few guesses are then generated based on the matchings derived by the first step. One balloon is initialized for every guess with an objective contour (generated from the shape model) associated with it. The inflating force for each balloon is adjusted to drive it towards the objective contour. (3) Steps (1) and (2) are repeated until no further balloons are initialized. The final shapes of the series of balloons are assessed, using the shape models, underlying image content, and other factors, and the best result is chosen as the final contour.

Appendix A

Dynamic Programming for Energy Minimization of Balloons

The Time-Delayed Discrete Dynamic Programming method was first used for energy minimization of ACMs by Amini *et al.* in [3]. In this appendix, we give a brief account of this method and an introduction of how it can be applied to minimize the energy of balloons.

Considering a snake with second-order internal energy, the total energy is:

$$E_{total} = \sum_{i=0}^{n-1} E_{internal}(v_i) + E_{external}(v_i) \tag{A.1}$$

where the internal energy is defined by:

$$E_{internal}(v_i) = (\alpha_i|v_i - v_{i-1}|^2 + \beta_i|v_{i+1} - 2v_i + v_{i-1}|^2)/2 \tag{A.2}$$

and the external energy is usually derived from the image content.

E_{total} can be divided into multiple items:

$$E_{total}(v_1, v_2, \ldots, v_n) = E_1(v_1, v_2, v_3) + E_2(v_2, v_3, v_4) + \ldots + E_{n-2}(v_{n-2}, v_{n-1}, v_n) \tag{A.3}$$

where

$$E_{i-1}(v_{i-1}, v_i, v_{i+1}) = E_{external}(v_i) + E_{internal}(v_{i-1}, v_i, v_{i+1}) \tag{A.4}$$

To apply dynamic programming, the energy has to be minimized in stages. For each snaxel, a 3x3 grid is searched for possible moves. Searching for the best move for the whole snake starts from the first snaxel, then the second one, and so on. Each step is based on the results found by the previous step. A state table is used to store the decisions made by the previous step. For a snake with second-order internal energy involving a snaxel's two immediate neighbors, a vector of two variables is enough for each stage. At stage i,

$$s_i(v_{i+1}, v_i) = \min_{v_{i-1}} s_{i-1}(v_i, v_{i-1}) + \alpha|v_i - v_{i-1}|^2 + \beta|v_{i+1} - 2v_i + v_{i-1}|^2 + e_{external}(v_i) \tag{A.5}$$

where $s_i(v_{i+1}, v_i)$ is the state variable for stage i which involves three snaxels, v_{i+1}, v_i, and v_{i-1}. The state variable of this stage records the energies for all possible next-position combinations of snaxels v_{i+1} and v_i. This state variable is then used to compute the state variable at the next stage. The best move can be backtracked when reaching the final stage.

The time complexity of this algorithm, using a second-order internal energy model, is $O(nm^3)$, where n is the length of the snake and m=9, the size of a 3x3 neighborhood that is searched for each snaxel. The memory required for the state table is $O(nm^2)$. Now, if a snake's internal energy is computed at various scales, both the time complexity and storage requirement increase exponentially. For example, assuming that the internal energy of the partial balloon shown in Fig. A.1 is computed using 4 immediate neighbors of each snaxel, we have

$$E_{internal}(v_i) = (\alpha_i|v_i - v_{i-1}|^2 + \beta_i|v_{i+1} - 2v_i + v_{i-1}|^2)/2 + \rho_i|v_{i+2} - 2v_i + v_{i-2}|^2)/2 \quad (A.6)$$

The dynamic programming method is not directly applicable to the above energy formulation. Instead, we try to minimize the following internal energy:

$$E_{internal}(v_i) = (\alpha_i|v_{i-1} - v_{i-2}|^2 + \beta_i|v_i - 2v_{i-1} + v_{i-2}|^2)/2 + \rho_i|v_{i+2} - 2v_i + v_{i-2}|^2)/2 \quad (A.7)$$

The state variable is updated using:

$$\begin{aligned} s_{i+1}(v_{i+2}, v_{i+1}, v_i, v_{i-1}) = & \quad \min_{v_{i-2}} s_i(v_{i+1}, v_i, v_{i-1}, v_{i-2}) && (A.8) \\ + & \quad \alpha|v_{i-1} - v_{i-2}|^2 + \beta|v_i - 2v_{i-1} + v_{i-2}|^2 && (A.9) \\ + & \quad \rho|v_{i+2} - 2v_i + v_{i-2}|^2 + e_{external}(v_{i+1}) && (A.10) \end{aligned}$$

The time complexity increases from nm^3 to nm^5, and the storage requirement is now nm^4.

To minimize the energy of balloons, the inflating force has to be converted into a potential energy so that it can be used with other external energies. In Fig. A.2, for the normal inflating force **n**, the amount of potential energy for each of x's 8 neighbors is computed using:

$$E_{inflating} = -\frac{1}{2}[\bar{\mathbf{n}} \cdot \bar{\mathbf{w}}] \left|\frac{\mathbf{n}}{h}\right|^2, \quad (A.11)$$

where **w** is a vector from x to each of its neighbors. Barred vectors are unit vectors.

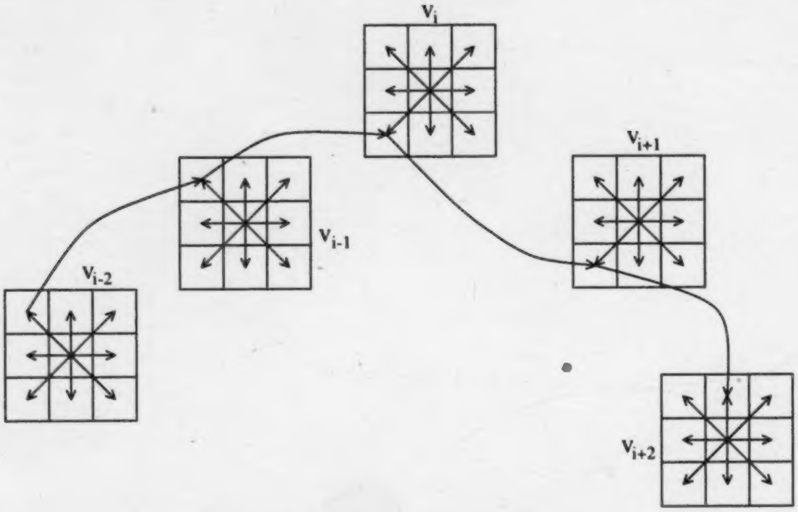


Figure A.1: The 3x3 grid (m=9) indicates the searching area for each snaxel. The number of combinations of possible moves involving 5 snaxels is m^5. Curved arrows show one possible move.

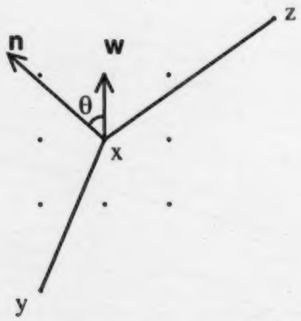


Figure A.2: Converting the normal inflating force into potential energies.

Appendix B

Proof of Snake Properties

LEMMA 1: The new internal energy, $E_{internal}$, as defined in Eqn. 3.6 is scale, rotation, and translation invariant.

Proof: A translation, rotation and scale transformation is defined as,

$$\mathbf{x}' = S\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}\mathbf{x} + \begin{pmatrix} x1_{offset} \\ x2_{offset} \end{pmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B} \tag{B.1}$$

From Eqn. 3.4, we have:

$$\begin{aligned}
r' &= \tfrac{|x'-y'|}{2\sin\theta} = \tfrac{|\mathbf{A}||x-y|}{2\sin\theta} && \text{(B.2)}\\
c_1' &= \tfrac{1}{2}(x'+y') + r'\sin\theta\mathbf{RU}(x'-y') && \text{(B.3)}\\
&= \tfrac{1}{2}(\mathbf{A}(x+y)+2\mathbf{B}) + \tfrac{|A||x-y|}{2\sin\theta}\sin\theta\mathbf{RU}(\mathbf{A}(x-y)) && \text{(B.4)}\\
&= \tfrac{1}{2}(\mathbf{A}(x+y)+2\mathbf{B}) + \tfrac{1}{2}|\mathbf{A}||x-y|\tfrac{\mathbf{A}}{|\mathbf{A}|}\mathbf{RU}(x-y) && \text{(B.5)}\\
&= \tfrac{1}{2}(\mathbf{A}(x+y)+2\mathbf{B}) + \tfrac{1}{2}\mathbf{A}|x-y|\mathbf{RU}(x-y) && \text{(B.6)}
\end{aligned}$$

Then, $c_1' - z'$ is derived:

$$\begin{aligned}
c_1' - z' &= \tfrac{1}{2}(\mathbf{A}(x+y)+2\mathbf{B}) + \tfrac{1}{2}\mathbf{A}|x-y|\mathbf{RU}(x-y) - (\mathbf{A}z - \mathbf{B}) && \text{(B.7)}\\
&= \tfrac{1}{2}\mathbf{A}((x+y-z) - |x-y|\mathbf{RU}(x-y)) && \text{(B.8)}\\
&= \mathbf{A}(c_1 - z) && \text{(B.9)}\\
|\mathbf{v}'| &= |(|c_1'-z'| - r')\mathbf{U}(c_1'-z')| && \text{(B.10)}\\
&= |\mathbf{A}|(c_1 - z - \tfrac{|x-y|}{2\sin\theta})|\mathbf{UA}(c_1-z)| && \text{(B.11)}\\
&= |\mathbf{A}|(c_1 - z - \tfrac{|x-y|}{2\sin\theta})|\mathbf{U}(c_1-z)| && \text{(B.12)}\\
&= |\mathbf{A}||\mathbf{v}| && \text{(B.13)}
\end{aligned}$$

Similarly, the following equation can be derived:

$$|\mathbf{w}'| = |\mathbf{A}||\mathbf{w}| \tag{B.14}$$

In Eqn. 3.6, h stands for the average distance between snaxels. After the transformation, we have:

$$h' = |\mathbf{A}|h \tag{B.15}$$

Therefore,

$$\begin{aligned}
E'_{internal} &= \frac{1}{2}\frac{|\mathbf{v}'|^2+|\mathbf{w}'|^2}{(h')^2} && \text{(B.16)}\\
&= \frac{1}{2}\frac{|\mathbf{A}|^2(|\mathbf{v}|^2+|\mathbf{w}|^2)}{|\mathbf{A}|^2h^2} && \text{(B.17)}\\
&= \frac{1}{2}\frac{|\mathbf{v}|^2+|\mathbf{w}|^2}{(h)^2} && \text{(B.18)}\\
&= E_{internal} && \text{(B.19)}
\end{aligned}$$

Hence, $E_{internal}$ is scale, rotation, and translation invariant.

Bibliography

[1] T. D. Alter and R. Basri. Extracting salient curves from images: An analysis of the saliency network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 13–20, 1996.

[2] A. A. Amini, S. Tehrani, and T. E. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *Proceedings of Int'l Conference on Computer Vision*, pages 95–99, 1988.

[3] A. A. Amini, T. E. Weymouth, and R. C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[4] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13:111–122, 1981.

[5] M.-O. Berger and R. Mohr. Towards autonomy in active contour models. In *Proceedings of Int'l Conference on Pattern Recognition*, pages 847–851, 1990.

[6] M. Bertero, T. Poggio, and V. Torre. Ill-posed problems in early vision. Memo 924, MIT A.I. Lab, 1987.

[7] A. C. Bovik. On detecting edges in speckle imagery. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(10):1618–1627, 1988.

[8] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):679–698, 1986.

[9] J. F. Canny. Finding edges and lines in images. Technical Report 720, MIT A.I. Lab, 1983.

[10] A. Chakraborty, L. H. Staib, and J. S. Duncan. Deformable boundary finding in medical images by integrating gradient and region information. *IEEE Transactions on Medical Imaging*, 15(6):859–870, 1996.

[11] A. Chakraborty, M. Worring, and J. S. Duncan. On multi-feature integration for deformable boundary finding. In *Proceedings of Int'l Conference on Computer Vision*, pages 846–851, 1995.

[12] V. Chalana, D. T. Linker, D. R. Haynor, and Y. Kim. A multiple active contour model for cardiac boundary detection on echocardiographics sequences. *IEEE Transactions on Medical Imaging*, 15(3):290–298, 1996.

[13] G. I Chiou and J.-N. Hwang. A neural network-based stochastic active contour model (NNS-SNAKE) for contour finding of distinct features. *IEEE Transactions on Image Processing*, 4(10):1407–1416, 1995.

[14] L. D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-D and 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, 1993.

[15] T. F. Cootes and C. J. Taylor. Active shape models: Smart snakes. In *Proceedings of British Machine Vision Conf.*, pages 266–275, 1992.

[16] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proceedings of British Machine Vision Conf.*, pages 9–18, 1992.

[17] R. N. Czerwinski, D. L. Jones, and W. D. O'Brien Jr. Edge detection in ultrasound speckle noise. In *Proceedings of Int'l Conference on Image Processing*, volume 3, pages 304–308, 1994.

[18] J. S. Duncan and T. Birkholzer. Reinforcement of linear structure using parametrized relaxation labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(5):502–515, 1992.

[19] N. D. Efford. Knowledge-based segmentation and feature analysis of hand-wrist radiographs. Research Report 94.31, School of Computer Studies, University of Leeds, 1994.

[20] K. S. Fu and J. K. Mui. A survey on image segmentation. *Pattern Recognition*, 13:3–16, 1981.

[21] P. Fua and Y. G. Leclerc. Model driven edge detection. *Machine Vision and Applications*, 3:45–56, 1990.

[22] R. C. Gonzalez and P. Wintz. *Digital Image Processing, Second Edition.* Addison-Wesley Publishing Company, 1987.

[23] S. R. Gunn and M. S. Nixon. A robust snake implementation; A dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):63–68, 1997.

[24] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *Int'l Journal of Computer Vision*, 20(1/2):113–133, 1996.

[25] R. Haralick and L. Shapiro. *Computer and Robot Vision, I II.* Addison Wesley, 1992.

[26] R. M Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, 1984.

[27] R. M. Haralick and L. G. Shapiro. Survey: Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29:133–139, 1985.

[28] J. Ivins and J. Porrill. Active region models for segmenting textures and colours. *Image and Vision Computing*, 13(5):1995, 431–438.

[29] J. Jones and L. Palmer. The two-dimensional spatial structure of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58:1187–1211, 1987.

[30] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *Proceedings of Int'l Conference on Computer Vision*, pages 259–268, 1987.

[31] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int'l Journal of Computer Vision*, 1(4):321–331, 1988.

[32] K. F. Lai. *Deformable Contours: Modeling, Extraction, Detection and Classification*. PhD thesis, University of Wisconsin-Madison, 1994.

[33] F. Leymarie and M. D. Levine. Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):617–643, 1993.

[34] X. Li and J. Wang. Adaptive balloon models. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 434–439, 1999.

[35] D. Marr. On the purpose of low-level vision. Memo 324, MIT A.I. Lab, 1974.

[36] A. Martelli. An application of heuristic search methods to edge and contour detection. *Communications of ACM*, 2:73–83, 1976.

[37] V. S. Nalwa and E. Pauchon. Edgel aggregation and edge description. *Computer Graphics and Image Processing*, 40:79–94, 1987.

[38] W. M. Neuenschwander, P. Fua, L. Iverson, G. Szekely, and O. Kubler. Ziplock snakes. *Int'l Journal of Computer Vision*, 25(3):191–201, 1997.

[39] B. Olstad and A. H. Torp. Encoding of A priori information in active contour models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):863–872, 1996.

[40] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9):1277–1294, 1993.

[41] E. Persoon and K. S. Fu. Shape discrimination using Fourier Descriptors. *IEEE Trans. Systems, Man, and Cybernetics*, 7(3):170–179, 1977.

[42] P. Radeva, J. Serrat, and E. Marti. A snake for model-based segmentation. In *Proceedings of Int'l Conference on Computer Vision*, pages 816–821, 1995.

[43] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.

[44] A. Shaashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proceedings of Int'l Conference on Computer Vision*, pages 321–327, 1988.

[45] L. H. Staib and J. S. Duncan. Boundary finding with parametrically deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:1061–1075, 1992.

[46] L. H. Staib and J. S. Duncan. Model-based deformable surface finding for medical images. *IEEE Transactions on Medical Imaging*, 15(5):720–731, 1996.

[47] G. Storvik. A Bayesian approach to dynamic contours through stochastic sampling and simulated annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):976–986, 1994.

[48] V. Torre and T. A. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):147–163, 1986.

[49] J. Wang. A content-guided searching algorithm for balloons. In *Proceedings of the Fifth International Conference for Young Computer Scientists*, volume II, pages 1072–1076, 1999.

[50] J. Wang and X. Li. Guiding ziplock snakes with a priori information. *To appear in Proceedings of ICPR '00.*

[51] J. Wang and X. Li. A system for segmenting ultrasound images. In *Proceedings of ICPR '98*, pages 456–461, 1998.

[52] J. Wang and X. Li. A self-adjusting mechanism for active contour models. In *Proceedings of Vision Interface '00*, pages 31–37, 2000.

[53] J. Wang, X. Li, and A. Bradley. Boundary searching snakes for segmenting noisy images. In *Proceedings of Vision Interface '98*, pages 107–114, 1998.

[54] D. J. Williams and M. Shah. A fast algorithm for active contours. In *Proceedings of Int'l Conference on Computer Vision*, pages 592–595, 1990.

[55] A. Witkin. Scale space filtering: A new approach to multiscale description. In S. Ullman and W. Richards, editors, *Image Understanding*, pages 79–95. Ablex Publishing Corporation, 1984.

[56] M. Worring, A. W. M. Smeulders, and L. H. Staib J. S. Duncan. Parameterized feasible boundaries in gradient vector fields. In *Proceedings of 13th International Conference on Information Processing in Medical Imaging*, pages 48–61. Springer-Verlag, 1993.

[57] G. Xu, E. Segawa, and S. Tsuji. Robust active contours with insensitive parameters. *Pattern Recognition*, 27(7):879–884, 1994.

[58] Q. Zhu, M. Payne, and V. Riordan. Edge linking by a directional potential function(DPF). *Image and Vision Computing*, 14:59–70, 1996.